

ESCUELA TÉCNICA SUPERIOR DE NÁUTICA
UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Grado

**SIMULACIÓN DE PROCEDIMIENTOS DE
CARGA EN BUQUES
PORTACONTENEDORES CON JAVA**

**STOWAGE PROCEDUREMENTS
SIMULATION FOR CONTAINER SHIPS WITH
JAVA**

Para acceder al Título de Grado en

**INGENIERÍA NÁUTICA Y TRANSPORTE
MARÍTIMO**

Autor: Luis Moreno Fernández

Director: Francisco J. Correa Ruiz

Septiembre - 2021

ESCUELA TÉCNICA SUPERIOR DE NÁUTICA
UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

**SIMULACIÓN DE PROCEDIMIENTOS DE
CARGA EN BUQUES
PORTACONTENEDORES CON JAVA**

**STOWAGE PROCEDUREMENTS
SIMULATION FOR CONTAINER SHIPS WITH
JAVA**

Para acceder al Título de Grado en

**INGENIERÍA NÁUTICA Y TRANSPORTE
MARÍTIMO**

Septiembre - 2021

ÍNDICE

RESUMEN	5
PALABRAS CLAVE	5
ABSTRACT	6
KEY WORDS.....	6
LISTA DE FIGURAS	7
1. INTRODUCCIÓN.....	9
2. OBJETIVOS	10
3. METODOLOGÍA	11
4. ESTRUCTURA.....	11
4.1. PAQUETE CLASES.....	14
4.1.1. CLASE FECHA	15
4.1.2. CLASE EMPRESA.....	17
4.1.3. CLASE PUERTO	19
4.1.4. CLASE MMPP	21
4.1.5. CLASE ESCOTILLA	30
4.1.6. CLASE CONTENEDOR.....	31
4.1.7. CLASE BARCO.....	36
4.1.8. CLASE RUTA	46
4.1.9. CLASE VIAJE	48
4.2. PAQUETE IMDG	50
4.3. PAQUETE ASTILLERO.....	53
Ejemplo: Buque Beatriz B	54
4.4. PAQUETE SOPORTE	60
4.4.1 Generación de contenedores	62
4.5. PAQUETE INTERFAZ	68
4.5.1. CLASE ESTADÍSTICAS	69
4.5.2. CLASE DIBUJO	75
5. ALGORITMOS DE CARGA	80
5.1. VERSIONES	81
5.2. SEGREGACIÓN DE MMPP.....	95
6. INTERFACES Y FUNCIONAMIENTO	99
7. APLICACIÓN PRÁCTICA DE SIGECO.....	104
Ejemplo: Muestra del funcionamiento los algoritmos	104
Ejemplo A: Beatriz B y una ruta Mediterránea.	107
Ejemplo B: MSC Positano y una ruta por África.	112

Ejemplo C: Caroline Maersk y una ruta entre Europa y Extremo Oriente.....	115
8. OBSERVACIONES DEL PROGRAMA.....	118
9. CONCLUSIONES.....	120
ANEXO I: INDICACIONES SOBRE JAVA.....	121
ANEXO II: AVISO RESPONSABILIDAD UC	128
BIBLIOGRAFÍA	129

RESUMEN

En este trabajo se explican las bases para desarrollar un sencillo programa de estiba para buques portacontenedores que pretende simular y evaluar diferentes formas de realizar este tipo de procedimientos. Para la elaboración de este proyecto se escribió un programa funcional en java. La finalidad de éste es demostrar que los conceptos que se describen aquí tienen aplicación real. Del programa se mostrarán secciones del código y fotografías de sus interfaces.

PALABRAS CLAVE

Java

Programa

Estiba

Contenedor

Segregación

ABSTRACT

This essay explains the basics for the development a simple program of stowage for container vessels that aims to simulate and assess different ways of make this type of procedurements. For the development of this project a functional program was coded in java. It's purpose is prove that the concepts that are described here have a real aplicattion. Code excerpts and photographs of it's interfaces will be displayed.

KEY WORDS

Java

Program

Stowage

Container

Segregation

LISTA DE FIGURAS

ILUSTRACIÓN 1: ESQUEMA DE LA ESTRUCTURA DEL PROGRAMA. FUENTE: PROPIA.....	13
ILUSTRACIÓN 2: CUADRO DE SEGREGACIÓN. FUENTE: CÓDIGO IMDG:.....	26
ILUSTRACIÓN 3: CUADRO PARA LA ESTIBA MIXTA. FUENTE: CÓDIGO IMDG.....	27
ILUSTRACIÓN 4: DIAGRAMA DE BAHÍAS DUPLAS (COLOREADAS EN VERDE O BLANCO). FUENTE: PROPIA.	38
ILUSTRACIÓN 5: DIAGRAMA DE BAHÍAS SIMPLES (COLOREADAS EN AZUL VERDOSO O BLANCO). FUENTE: PROPIA.....	38
ILUSTRACIÓN 6: DIAGRAMA DE CUBIERTA/BODEGA (COLOREADAS EN AZUL O BLANCO). FUENTE: PROPIA.	39
ILUSTRACIÓN 7: DIAGRAMA DE FILAS (COLOREADAS EN AZUL OSCURO O BLANCO). FUENTE: PROPIA.....	39
ILUSTRACIÓN 8: DIAGRAMA DE ALTURAS (COLOREADAS EN ROJO O BLANCO). FUENTE: PROPIA.	40
ILUSTRACIÓN 9: PROA DE UN PORTACONTENEDORES. VÉANSE TRES CONTENEDORES COLOREADOS. FUENTE: PROPIA.	40
ILUSTRACIÓN 10: EL BUQUE BEATRIZ B. FUENTE: BOLUDA CORPORACIÓN MARÍTIMA.	54
ILUSTRACIÓN 11: BUQUE BEATRIZ B EN MODO TRANSVERSAL. FUENTE: PROPIA.....	75
ILUSTRACIÓN 12: BAHÍA DUPLA 10. COMPUESTA POR DOS SIMPLES 9 Y 11, FUENTE: PROPIA.	75
ILUSTRACIÓN 13: BUQUE BEATRIZ B EN MODO LONGITUDINAL, FILA 1. FUENTE: PROPIA.....	76
ILUSTRACIÓN 14: BAHÍA 10, FILA 1. FUENTE: PROPIA.	76
ILUSTRACIÓN 15: LEYENDA DE DIMENSIONES. FUENTE: PROPIA.....	76
ILUSTRACIÓN 16: LEYENDA DE TIPO. FUENTE: PROPIA.	77
ILUSTRACIÓN 17: LEYENDA DEL MTY. FUENTE: PROPIA.	77
ILUSTRACIÓN 18: LEYENDA DE PESAJES. FUENTE: PROPIA.	77
ILUSTRACIÓN 19: LEYENDA DE MMPP. FUENTE: PROPIA.....	77
ILUSTRACIÓN 20: LEYENDA DE EMPRESAS, PARTE 1. FUENTE: PROPIA.....	78
ILUSTRACIÓN 21: LEYENDA DE EMPRESAS, PARTE 2. FUENTE: PROPIA.....	78
ILUSTRACIÓN 22: LEYENDA DE CONTENEDORES CÓDIGO. FUENTE: PROPIA.	79
ILUSTRACIÓN 23: CUADRO DE ESTADÍSTICAS DEL BEATRIZ B ESTANDO VACÍO. FUENTE: PROPIA.	79
ILUSTRACIÓN 24: ESTRUCTURA DE UN ALGORITMO DE CARGA. FUENTE: PROPIA.....	80
ILUSTRACIÓN 25: DIAGRAMA-CUESTIONARIO DE SEGREGACIÓN. FUENTE: CÓDIGO IMDG.	95
ILUSTRACIÓN 26: CUADRO DE SEGREGACIÓN, APARTADO 7.4.3.2 FUENTE: CÓDIGO IMDG.	96
ILUSTRACIÓN 27: PANEL PARA LA GENERACIÓN DE UN VIAJE. FUENTE: PROPIA.....	99
ILUSTRACIÓN 28: MENÚ GENERAR RUTA, PARTE 1. FUENTE: PROPIA.....	100
ILUSTRACIÓN 29: MENÚ GENERAR RUTA, PARTE 1. FUENTE: PROPIA.....	100
ILUSTRACIÓN 30: MENÚ GENERAR RUTA, PARTE 2. FUENTE: PROPIA.....	101
ILUSTRACIÓN 31: MENÚ GENERAR RUTA, PARTE 2. FUENTE: PROPIA.....	101
ILUSTRACIÓN 32: INTERFAZ PRINCIPAL DE SIGECO. FUENTE: PROPIA.....	103
ILUSTRACIÓN 33: SIGECO PARA BEATRIZ B CON CARGABASICA_v0. FUENTE: PROPIA.	104
ILUSTRACIÓN 34: SIGECO PARA BEATRIZ B CON CARGABASICA_v0. FUENTE: PROPIA.	105
ILUSTRACIÓN 35: SIGECO PARA BEATRIZ B CON CARGABASICA_v0. FUENTE: PROPIA.	105
ILUSTRACIÓN 36: SIGECO PARA BEATRIZ B CON CARGABASICA_v3. FUENTE: PROPIA.	106
ILUSTRACIÓN 37: SIGECO PARA BEATRIZ B CON CARGABASICA_v3. FUENTE: PROPIA.	106
ILUSTRACIÓN 38: SIGECO PARA BEATRIZ B CON CARGABASICA_v3. FUENTE: PROPIA.	107
ILUSTRACIÓN 39: EL BUQUE BEATRIZ B. FUENTE: MARINETRAFFIC, AUTOR: CELSO HERNÁNDEZ.....	107
ILUSTRACIÓN 40: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	108
ILUSTRACIÓN 41: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	108
ILUSTRACIÓN 42: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	109
ILUSTRACIÓN 43: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	109
ILUSTRACIÓN 44: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	110
ILUSTRACIÓN 45: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	110
ILUSTRACIÓN 46: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	111
ILUSTRACIÓN 47: SIGECO PARA BEATRIZ B Y RUTA MEDITERRÁNEA. FUENTE: PROPIA.....	111
ILUSTRACIÓN 48: EL BUQUE MSC POSITANO. FUENTE: MARINETRAFFIC. AUTOR: DAVID LEONARD.....	112
ILUSTRACIÓN 49: SIGECO PARA MSC POSITANO Y RUTA AFRICANA. FUENTE: PROPIA.....	112
ILUSTRACIÓN 50: SIGECO PARA MSC POSITANO Y RUTA AFRICANA. FUENTE: PROPIA.....	113
ILUSTRACIÓN 51: SIGECO PARA MSC POSITANO Y RUTA AFRICANA. FUENTE: PROPIA.....	113
ILUSTRACIÓN 52: SIGECO PARA MSC POSITANO Y RUTA AFRICANA. FUENTE: PROPIA.....	114

ILUSTRACIÓN 53: SIGECO PARA MSC POSITANO Y RUTA AFRICANA. FUENTE: PROPIA.....	114
ILUSTRACIÓN 54: EL BUQUE CAROLINE MAERSK. FUENTE: MARINETRAFFIC. AUTOR: SERGEI SKRIABIN.....	115
ILUSTRACIÓN 55: SIGECO PARA CAROLINE MAERSK Y RUTA ASIÁTICA. FUENTE: PROPIA.	115
ILUSTRACIÓN 56: SIGECO PARA CAROLINE MAERSK Y RUTA ASIÁTICA. FUENTE: PROPIA.	116
ILUSTRACIÓN 57: SIGECO PARA CAROLINE MAERSK Y RUTA ASIÁTICA. FUENTE: PROPIA.	116
ILUSTRACIÓN 58: SIGECO PARA CAROLINE MAERSK Y RUTA ASIÁTICA. FUENTE: PROPIA.	117
ILUSTRACIÓN 59: SIGECO PARA CAROLINE MAERSK Y RUTA ASIÁTICA. FUENTE: PROPIA.	117

1. INTRODUCCIÓN

A mediados del siglo XX Malcolm McLean, un transportista estadounidense, observó y dedujo que el modo de transportar las mercancías en fardos individuales era costoso e ineficiente. En 1953 decidió montar una empresa que transportaba camiones sobre buques, esta idea poco a poco fue derivando hasta transportar únicamente la caja del camión y estandarizar las dimensiones de esta, dando lugar al contenedor que conocemos hoy. Por aquel entonces ya existían contenedores marítimos, pero fue Malcolm el que tuvo la visión de que tanto el buque, la terminal y la logística involucrada debían de adaptarse para hacer eficientes y económicas las operaciones.

En 1956 el buque Ideal-X se convierte en el pionero del transporte containerizado llevando 58 contenedores. Desde entonces, el número de TEU (unidad de contenedor correspondiente a 20 pies), no ha hecho más que crecer. En el año de 2007 Maersk pone en circulación la clase triple E, con 11.000 TEU. En el año 2021, en el momento de la elaboración de este trabajo, la naviera Evergreen está construyendo una nueva clase de portacontenedores, el primero de los cuales es el Ever Ace, de 24.000 TEU.

Para mantener tales flujos en funcionamiento hay detrás un entramado logístico cimentado en potentes softwares encargados de gestionar y organizar estos contenedores en puertos, almacenes y buques. De esta manera se pueden ir planificando las estibas en tiempo real y gracias a ello se sostienen los grandes volúmenes de tráfico. Dichos programas son de uso privado y de difícil acceso.

Con este proyecto podremos proporcionar un marco que pretende emular dichos programas, en el que dados una serie de puertos generaremos una serie de listados de contenedores. Un barco recorrerá cada uno de estos puertos y se cargará en base a estas listas. Podrá analizar múltiples casos en los que haya varios tipos de carga y estudiar sus resultados.

Estudiaremos la estructura del código, las partes que lo componen y como interaccionan entre ellas. Por último, se expondrán tres ejemplos: En cada uno de ellos habrá expuestos tres barcos distintos y tres rutas por el Mediterráneo, África y de Asia a Europa.

Para la elaboración de este código nos valdremos de Java. Al final de este trabajo, en el anexo, se adjuntará una breve introducción a este lenguaje de programación.

2. OBJETIVOS

Las metas planteadas en este trabajo son las expuestas:

- Crear un entorno virtual en el que dado un barco y una lista de contenedores se podrá realizar una carga.
- Elaboración de algoritmos de posicionamiento en una matriz tridimensional de contenedores. Estos algoritmos podrán descartar, cargar y descargar contenedores. Los criterios básicos son:
 - Distribución del barco.
 - Dimensiones de los contenedores.
 - Peso.
 - Puerto de salida.
 - Presencia de mercancías peligrosas.
- Generación de planos de las bahías para cada una de las cargas.
- Elaboración de cuadros de estadísticas y listados de los contenedores cargados.
- El software podrá trabajar con distintos tipos de barco, listas y rutas. Los algoritmos se tendrán que adaptar a sus variaciones.
- Para la simulación de múltiples casos se creará adicionalmente un generador de listas de contenedores realista.

3. METODOLOGÍA

Para la elaboración del código se partió de con un planteamiento general sobre cómo debía de estructurarse el programa, así como de una serie de dibujos y esbozos de cómo sería la interfaz principal. Conforme se iban obteniendo los primeros resultados se aplicaban las correcciones pertinentes.

Por ejemplo, durante el desarrollo se habían escrito los planos de los barcos por medio de clases y al cabo de un tiempo, se cambió de opción para en su lugar establecer una matriz de cinco dimensiones por ser más sencillo de escribir y recorrer. De igual manera, cuando llegó el momento de escribir las funciones de llenado de los barcos, se partió de un algoritmo muy básico que llenaba los barcos sin atender a criterios y para cada una de las versiones posteriores de esta se le iban añadiendo más funcionalidades.

Se ha buscado la eficiencia, la funcionalidad y la sencillez. Este es un trabajo elaborado de una forma heurística.

4. ESTRUCTURA

El programa se llama SIGECO (Sistema Gestión de Contenedores). Ha sido escrito en el entorno de desarrollo (IDE) Netbeans.

El programa necesita tres parámetros de entrada (inputs):

- Un barco (puede venir con contenedores o estar vacío), se llama desde el paquete astillero.
- Una ruta (serie de puertos en los que en cada uno habrá un packing list). Se crean por medio del generador de rutas del paquete soporte. Una vez guardados en el ordenador se pueden cargar desde el explorador de archivos.
- El tipo de algoritmo de carga. Están alojados en la clase viaje.

El programa devolverá (outputs) una serie de dibujos con el plan de carga del barco y un cuadro con las estadísticas con datos como los pesos o número de contenedores.

Está compuesto por cinco paquetes:

- Paquete clases: En éste se encuentra el código básico del programa.
- Paquete soporte: Aquí se alojan las funciones de generación de listas de contenedores y se guarda la información relacionada con puertos y empresas.
- Paquete imdg: En este paquete se guarda la información relacionada con el código imdg.
- Paquete astillero: Aquí se encuentran las clases que guardan los objetos barco.
- Paquete interfaz: Aquí se encuentra el código relacionado con los dibujos, los menús (hechos con las librerías de Oracle JFrame) y el almacenamiento de la información. El método main() se encuentra aquí.

Como se muestra en la ilustración 1 se puede observar un esquema simplificado de como interaccionan las clases entre ellas:

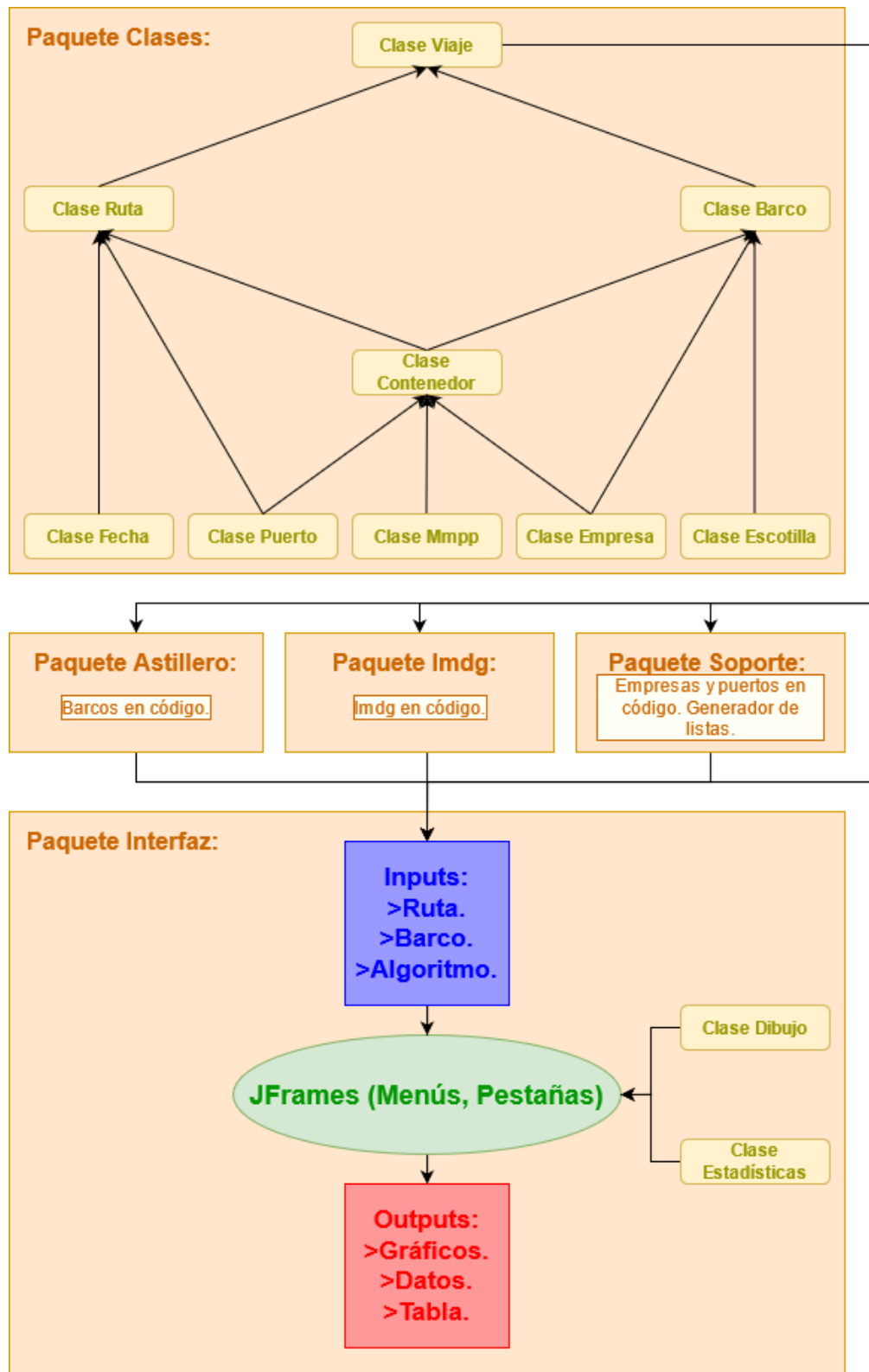


Ilustración 1: Esquema de la estructura del programa. FUENTE: Propia.

4.1. PAQUETE CLASES

En este paquete hay nueve clases que abordan aspectos básicos del funcionamiento del programa. En cada una de estas clases se codifica un objeto: Por lo que en este paquete solo se alojan constructores y demás métodos asociados al funcionamiento de estos objetos.

- Clase **Fecha**: Se utiliza en la clase **Ruta**, pero actualmente tiene una finalidad pormenor.
- Clase **Empresa**: Esta clase es utilizada en **Contenedor** y el **Barco** para indicar a que empresa pertenecen. Es importante para la función de generación de listas puesto que dependiendo de la empresa que sea el contenedor, variará el tipo de contenedores que pueda tener. Por ejemplo, la empresa Tankcon International solo generará contenedores cisterna.
- Clase **Puerto**: Necesaria para saber las localizaciones por las que se pasará en la clase **Ruta** y los POL/POD de la clase **Contenedor**.
- Clase **Mmpp**: Indica la presencia de una mercancía peligrosa en la clase **Contenedor** y los detalles de ésta.
- Clase **Escotilla**: Es un objeto dedicado a comprobar si las escotillas del **Barco** se encuentran disponibles.
- Clase **Contenedor**: Los contenedores se encontrarán en los packing lists de cada uno de los puertos de la clase **Ruta** y contenidos en las bahías de la clase **Barco**.
- Clase **Barco**: El vehículo que se va a ir modificando a lo largo de la clase **Viaje**.
- Clase **Ruta**: El tránsito que se va a realizar en la clase **Viaje**.
- Clase **Viaje**: En esta clase se reúnen todas las clases y es sobre la que el usuario solicitará los datos de los inputs.

4.1.1. CLASE FECHA

Uso.

El objeto **Fecha** se creó con vistas a que pudiese ser necesario realizar un informe sobre el viaje en el que resultase conveniente indicar los tiempos de tránsito, también se hizo en el que caso de que fuesen necesarios cálculos del consumo del barco. Actualmente, esta clase tiene una finalidad informativa.

Esta clase importa la librería de Oracle `Serializable`, necesaria para el guardado de la información.

Declaración de atributos.

```
1 int hora;  
2 int minuto;  
3 int segundo;  
4 int dia;
```

En este caso, los datos son muy intuitivos. Cuenta con cuatro atributos numéricos enteros.

Constructor.

```
1 public Fecha() {}  
2  
3 public Fecha(int hora, int minuto, int segundo, int dia) {  
4     this.hora = hora;  
5     this.minuto = minuto;  
6     this.segundo = segundo;  
7     this.dia = dia;  
8 }
```

Métodos secundarios de interés.

Convendría reseñar el método `corrigeFecha`. Detecta que los datos introducidos en un objeto **Fecha** no superen sus rangos lógicos. De hacerlo, se resta lo que les sobra y se suma al dato superior:

```
1 public void corrigeFecha() {  
2     while (60 < segundo) {  
3         segundo = segundo - 60;
```

```

4         minuto = minuto + 1;
5     }
6     while (0 > segundo) {
7         segundo = segundo + 60;
8         minuto = minuto - 1;
9     }
10    while (60 < minuto) {
11        minuto = minuto - 60;
12        hora = hora + 1;
13    }
14    while (0 > minuto) {
15        minuto = minuto + 60;
16        hora = hora - 1;
17    }
18    while (24 < hora) {
19        hora = hora - 24;
20        dia = dia + 1;
21    }
22    while (0 > hora) {
23        dia = dia - 1;
24        hora = hora + 24;
25    }
26 }

```

Otro método de interés es `obtenerDecimal`, para el paso de un objeto **Fecha** a un valor decimal a `double`.

Para valores inferiores a 24h.

```

1 public double obtenerDecimal() {
2     return (double) hora + ((double) minuto / 60) + ((double) segundo /
3     3600);
4 }

```

El método `obtenerFechaDeDecimal`, es su inverso, realiza el paso de decimal en `double` a un objeto **Fecha**.

Para valores inferiores a 24h.

```

1 public void obtenerFechaDeDecimal(double dec) {
2     double hor, min, seg;
3     hor = (int) dec;
4     min = ((int) (dec - hor)) * 60;
5     seg = (dec - hor - (min / 60)) * 3600;
6     this.hora = (int) hor;
7     this.minuto = (int) min;
8     this.segundo = (int) seg;
9     this.dia = 0;
10    this.corrigeFecha();
11 }

```


4.1.2. CLASE EMPRESA

Uso.

El objeto **Empresa** tiene tres funciones en el código:

- En la generación de packing lists, dependiendo de la empresa, cada una tendrá un tipo de contenedores específico (dimensiones y uso), lo que aumentará el realismo de la generación. Además, habrá empresas que tengan una mayor probabilidad de aparecer que otras, por ejemplo, los contenedores de Hapag-Lloyd se generarán con más frecuencia que los de Hamburg Süd.
- En los códigos de propietario. Éstos forman parte del sistema de identificación de un contenedor. Dependiendo de la empresa, la matrícula del contenedor cambiará para ajustarse a su código.
- En los visores de contenedores, en la interfaz, hay un modo “empresa” que nos permite ver el plano de carga del barco coloreado en base a los colores corporativos de cada compañía, dando una idea aproximada de cómo sería el aspecto real de la pila de contenedores una vez cargada.

Esta clase importa las librerías de Oracle `Serializable` y `Color` necesarias para el guardado de la información y asignar colores identificativos en el visor, respectivamente.

Declaración de atributos.

```
1 String nombre;  
2 String bic;  
3 String pais;  
4 Color colorFuente;  
5 Color colorFondo;  
6 double porcen;  
7 char conhab;
```

- `nombre (String)`: Nombre genérico de la empresa.
- `bic (String)`: Código BIC, de tres letras correspondientes a la empresa asignadas por la Oficina Internacional de Contenedores.
- `pais (String)`: País o territorio dependiente de la empresa.

- `colorFuente` (`Color`): Color distintivo de la empresa, es el color del texto del contenedor, se usará cuando se muestre como texto en el visor leyenda de la interfaz.
- `colorFondo` (`Color`): Color distintivo de la empresa, se usa como color de fondo del contenedor, se usará cuando se muestre en la interfaz.
- `porcen` (`double`): Valor numérico que indica la probabilidad de generación de la empresa en las listas de contenedores.
- `conhab` (`char`): Es un carácter que indica el tipo de contenedores que posee la empresa. Los caracteres posibles son:
 - 'a': Contenedores de 20', y 40' de alturas ST y HC. De tipos G, V y B.
 - 'b': Contenedores de 45' + ('a').
 - 'c': Contenedores refrigerados de 40' HC y ST, y de 20' HC.
 - 'd': ('a') + ('c').
 - 'e': Contenedores tanque de 20', y 40' de alturas ST y HC.
 - 'f': ('b') + Nuevos subtipos.
 - 'g': ('f') + Contenedores HC y ST de 10'.
 - 'h': ('f') + Contenedores de 20' Half Height.
 - 'i': ('f') + Contenedores HC y ST de 10' + Contenedores de 20' Half Height.
 - 'j': ('d') + ('e')

Las categorías ('g'), ('h') e ('i') no están implementadas en el programa, las empresas que tengan estos valores se sustituirán por ('f').

Constructor.

```

1 public Empresa() {}
2
3 public Empresa(String nombre, String bic, String pais, Color colorFuente,
  Color colorFondo, double porcen, char conhab) {
4     this.nombre = nombre;
5     this.bic = bic;
6     this.pais = pais;
7     this.colorFuente = colorFuente;
8     this.colorFondo = colorFondo;
9     this.porcen = porcen;
10    this.conhab = conhab;
11 }

```

4.1.3. CLASE PUERTO

Uso.

El objeto **Puerto** es un pequeño grupo de tres `String` que se utiliza para:

- En **Contenedor**, es la variable para POL y POD.
- En **Ruta**, en forma de array unidimensional, es la lista de localizaciones que el buque recorrerá.

Esta clase importa las librerías de Oracle `Serializable` y `ArrayList` necesarias para el guardado de la información y el manejo de `arraysList`, (matrices con nº de artículos variable), respectivamente.

Declaración de atributos.

```
1 String nombre;  
2 String locode;  
3 String pais;
```

- `nombre (String)`: Nombre genérico del puerto, en castellano.
- `locode (String)`: Código para el transporte y comercio UN/LOCODE, se rige por la norma ISO 3166. Consta de cinco caracteres, los dos primeros corresponden con el país donde se localiza el puerto, los tres siguientes a la localización concreta del puerto.
- `pais (String)`: País o territorio dependiente donde se encuentra la empresa asentada.

Constructor.

```
1 public Puerto() {}  
2  
3 public Puerto(String nombre, String locode, String pais) {  
4     this.nombre = nombre;  
5     this.locode = locode;  
6     this.pais = pais;  
7 }
```

Métodos secundarios de interés.

El siguiente método se utiliza para, a partir de un de un ArrayList de Puerto, convertir estos a un arrayList de texto con los locode.

El resultado se utiliza para introducir la lista en el JComboBox "posición" del menú.

```
1 public String[] arrayListPuerto_A_arrayString (ArrayList <Puerto>
  listaPuertos) {
2     String[] listaVisitas = new String[listaPuertos.size()-1];
3     for (int i=0; i<listaPuertos.size()-1; i++) {
4         listaVisitas[i]=listaPuertos.get(i).getLocode();
5     }
6     return listaVisitas;
7 }
```

4.1.4. CLASE MMPP

Uso.

La clase Mmpp (Mercancías Peligrosas), se corresponde con los parámetros que introduce dicho código en su lista. Se utiliza para:

- En el objeto **Contenedor**, se encarga de indicar la presencia de mercancía IMO.

Esta clase importa la librería de Oracle `Serializable`, necesaria para el guardado de la información.

Para realizar una estadística de cadenas de texto en el algoritmo de segregación se han tomado los métodos `similarity` y `editDistance` de stackoverflow, esto quedará referenciado en la bibliografía.

Declaración de atributos.

```
1 int un_1;  
2 String noex_2;  
3 String clase_3;  
4 String[] riesgoSecundario_4;  
5 String embalaje_5;  
6 String[] eys_16a;  
7 String[] eys_16b;
```

- `un_1` (`int`): Número de cuatro dígitos. Se utiliza para identificar mercancías peligrosas.
- `noex_2` (`String`): El nombre, en mayúsculas, de la mercancía peligrosa. Puede ir acompañado con información adicional en minúsculas.
- `clase_3` (`String`): Indica la clase de mercancía peligrosa de acuerdo con la clasificación del apartado 2.0.1.1 del código IMDG.
- `riesgoSecundario_4` (`String[]`): En forma de array con dos espacios. El primero indica el número de clase de riesgo secundario. El segundo espacio es un identificador sobre si es un contaminante marino (presencia de la letra "P").
- `embalaje_5` (`String`): Grupo de embalaje.
- `eys_16a` (`String[]`): Estiba y manejo. Códigos especificados en los

apartados 7.1.5 y 7.1.6 del código IMDG. Compuestos por una categoría y entre cero o varios códigos de estiba y manipulación.

- eys_16b (String[]): Segregación. Códigos de segregación definidos en el apartado 7.2.8 del código IMDG.

Constructor.

```
1 public Mmpp() {}
2 public Mmpp(int un_1, String noex_2, String clase_3, String[]
  riesgoSecundario_4, String embalaje_5, String[] eys_16a, String[] eys_16b)
  {
3     this.un_1 = un_1;
4     this.noex_2 = noex_2;
5     this.clase_3 = clase_3;
6     this.riesgoSecundario_4 = riesgoSecundario_4;
7     this.embalaje_5 = embalaje_5;
8     this.eys_16a = eys_16a;
9     this.eys_16b = eys_16b;
10 }
```

Métodos secundarios de interés.

El siguiente es una función muy sencilla pero que ayuda a comprender la lógica de las funciones de segregación. Cada uno de los códigos de estiba y manejo es leído por el programa e interpretado para su posicionado en el barco. Su equivalente, para la interpretación escrita es la siguiente:

```
1 public String obtenerEstibaYManejo(String estibaYManejo) {
2     switch (estibaYManejo) {
3         case "Categoría 1":
4             return "En cubierta en unidades de transporte cerradas o
  bajo cubierta.";
5         case "Categoría 2":
6             return "En cubierta en unidades de transporte cerradas o
  bajo cubierta.";
7         case "Categoría 3":
8             return "En cubierta en unidades de transporte cerradas o
  bajo cubierta.";
9         case "Categoría 4":
10            return "En cubierta en unidades de transporte cerradas o
  bajo cubierta en unidades de transporte cerradas.";
11        case "Categoría 5":
12            return "En cubierta solamente en unidades de transporte
  cerradas.";
13        case "Categoría A":
14            return "En cubierta o bajo cubierta.";
15        case "Categoría B":
16            return "En cubierta o bajo cubierta.";
17        case "Categoría C":
```

```

18         return "En cubierta solamente.";
19     case "Categoría D":
20         return "En cubierta solamente.";
21     case "Categoría E":
22         return "En cubierta o bajo cubierta.";
23     case "SW1":
24         return "Protegido de las fuentes de calor.";
25     case "SW2":
26         return "Apartado de los lugares habituales.";
27     case "SW3":
28         return "Se transportará a temperatura regulada.";
29     case "SW4":
30         return "Se requiere ventilación de superficie que ayude a
eliminar los vapores de cualquier disolvente residual.";
31     case "SW5":
32         return "En caso de estiba bajo cubierta, en un espacio
ventilado mecánicamente.";
33     case "SW6":
34         return "Cuando se estiben bajo cubierta, la ventilación
mecánica se ajustará a las disposiciones de la regla II-2/19 (II-2/54) "
35         + "\n del Convenio SOLAS aplicables a los
líquidos inflamables cuyo punto de inflamación es inferior a 23 °C
(v.c.).";
36     case "SW7":
37         return "Conforme a lo aprobado por las autoridades
competentes de los países que participen en la expedición.";
38     case "SW8":
39         return "Se podrá exigir ventilación. Antes de efectuar la
operación de carga, habrá que tener en cuenta la posible necesidad de "
40         + "\n abrir las escotillas en caso de incendio
para obtener la máxima ventilación y de utilizar agua en una emergencia, "
41         + "\n con el consiguiente riesgo que supondría la
inundación de los espacios de carga para la estabilidad del buque.";
42     case "SW9":
43         return "Proporcióñese una buena ventilación por entre los
bultos si la carga va ensacada. Se recomienda la estiba en doble hilera. "
44         + "\n La ilustración que figura en 7.6.2.7.2.3
muestra la manera de proceder. Durante el viaje, se medirá la temperatura
con "
45         + "\n regularidad a distintas profundidades de la
bodega y se mantendrá un registro de dichas mediciones. Si la temperatura
de "
46         + "\n la carga supera la temperatura ambiente y
continúa aumentando, se interrumpirá la ventilación.";
47     case "SW10":
48         return "A menos que se lleven en unidades de transporte
cerradas, las balas irán debidamente cubiertas con lonas o cualquier medio
"
49         + "\n de protección análogo. Los espacios de
carga estarán limpios y secos, y sin rastros de aceite ni grasa. Las
caperuzas de "
50         + "\n los ventiladores que den a los espacios de
carga estarán provistas de pantallas para chispas. Todas las demás
aberturas, "
51         + "\n vías de entrada y escotillas que den a esos
espacios de carga estarán bien cerradas. Si se interrumpen temporalmente
las "

```

```

52         + "\n operaciones de carga y quedan las
    escotillas destapadas, se mantendrá una guardia contra incendios. Durante
    la carga o descarga, "
53         + "\n estará prohibido fumar en las proximidades,
    y se mantendrán los dispositivos contra incendios en condiciones de
    utilización inmediata.";
54     case "SW11":
55         return "Las unidades de transporte deberán resguardarse de
    la luz solar directa. Los bultos que vayan en las unidades de transporte "
56         + "\n deberán estibarse de manera tal que se
    permita la suficiente circulación de aire en toda la carga.";
57     case "SW12":
58         return "Teniendo en cuenta cualquier prescripción
    suplementaria especificada en los documentos de transporte.";
59     case "SW13":
60         return "Teniendo en cuenta cualquier prescripción
    suplementaria especificada en el certificado o los certificados de
    aprobación expedido "
61         + "\n por la autoridad competente.";
62     case "SW14":
63         return "Categoría A únicamente si se satisfacen las
    disposiciones especiales de estiba que figuran en 7.4.1.4 y 7.6.2.8.4.";
64     case "SW15":
65         return "En el caso de los bidones metálicos, categoría de
    estiba B.";
66     case "SW16":
67         return "En el caso de las cargas unitarias que vayan en
    unidades de transporte abiertas, categoría de estiba B.";
68     case "SW17":
69         return "Categoría E, en el caso de unidades de transporte
    cerradas y cajas paleta únicamente. Se podrá exigir ventilación. "
70         + "\n Antes de efectuar la operación de carga,
    habrá que tener en cuenta la posible necesidad de abrir las escotillas en
    caso "
71         + "\n de incendio para obtener la máxima
    ventilación y de utilizar agua en una emergencia, con el consiguiente
    riesgo que supondría "
72         + "\n la inundación de los espacios de carga para
    la estabilidad del buque.";
73     case "SW18":
74         return "Categoría A, cuando se transporte de conformidad
    con lo dispuesto en P650.";
75     case "SW19":
76         return "En el caso de las baterías transportadas de
    conformidad con lo dispuesto en las disposiciones especiales 376 o 377,
    categoría C, "
77         + "\n a menos que se transporten a bordo de
    buques que efectúan viajes internacionales cortos.";
78     case "SW20":
79         return "En el caso del nitrato de uranilo hexahidratado en
    solución, se aplica la categoría de estiba D.";
80     case "SW21":
81         return "En el caso del uranio metálico pirofórico y el
    torio metálico pirofórico, se aplica la categoría de estiba D.";
82     case "SW22":
83         return "En el caso de los AEROSOLES de capacidad máxima de
    1 l: categoría A. En el caso de los AEROSOLES de capacidad superior a 1 l:

```



```

"
84         + "\n categoría B. En el caso de los AEROSOL DE
DESECHO: categoría C, apartado de los lugares habitables.";
85     case "SW23":
86         return "Cuando se transporte en un contenedor para
graneles BK3, véanse 7.6.2.12 y 7.7.3.9.";
87     case "SW24":
88         return "Por lo que respecta a las disposiciones especiales
sobre estiba, véanse 7.4.1.3 y 7.6.2.7.2.";
89     case "SW25":
90         return "Por lo que respecta a las disposiciones especiales
sobre estiba, véase 7.6.2.7.3.";
91     case "SW26":
92         return "Por lo que respecta a las disposiciones especiales
sobre estiba, véanse 7.4.1.4 y 7.6.2.11.1.1.";
93     case "SW27":
94         return "Por lo que respecta a las disposiciones especiales
sobre estiba, véase 7.6.2.7.2.1.";
95     case "SW28":
96         return "Conforme a lo aprobado por la autoridad competente
del país de origen.";
97     case "SW29":
98         return "En el caso de los motores o la maquinaria que
contengan combustibles cuyo punto de inflamación sea igual o superior a 23
°C, "
99         + "\n categoría de estiba A.";
100    case "H1":
101        return "Manténgase lo mas seco posible.";
102    case "H2":
103        return "Manténgase lo más fresco posible.";
104    case "H3":
105        return "Durante el transporte, debería estibarse (o
mantenerse) en un lugar fresco y bien ventilado.";
106    case "H4":
107        return "Si es preciso limpiar los espacios de carga en el
mar, se debe hacer por un procedimiento que ofrezca, por lo menos, igual
s"
108        + "\n seguridad y con un equipo de la misma
eficacia que el que se utilizaría en un puerto. Mientras no se efectúen
las operaciones de limpieza, "
109        + "\n los espacios de carga en que se haya
transportado asbesto permanecerán cerrados, y estará prohibido entrar en
ellos.";
110    default:
111        return "-";
112    }
113 }

```

Como se puede observar, para cada uno de los parámetros `String` de `obtenerEstibaYManejo` se devuelve un texto detallando las restricciones y requerimientos exigidos.

El siguiente método devolverá la segregación a tomar dadas dos clases, este método se basa en lo dado por cuadro de segregación del capítulo 7.2.4 del código IMDG.

CLASE	1.1 1.2 1.5	1.3 1.6	1.4	2.1	2.2	2.3	3	4.1	4.2	4.3	5.1	5.2	6.1	6.2	7	8	9
Explosivos 1.1, 1.2, 1.5	-	-	-	4	2	2	4	4	4	4	4	4	2	4	2	4	X
Explosivos 1.3, 1.6	-	-	-	4	2	2	4	3	3	4	4	4	2	4	2	2	X
Explosivos 1.4	-	-	-	2	1	1	2	2	2	2	2	2	X	4	2	2	X
Gases inflamables 2.1	4	4	2	X	X	X	2	1	2	2	2	2	X	4	2	1	X
Gases no tóxicos, no inflamables 2.2	2	2	1	X	X	X	1	X	1	X	X	1	X	2	1	X	X
Gases tóxicos 2.3	2	2	1	X	X	X	2	X	2	X	X	2	X	2	1	X	X
Líquidos inflamables 3	4	4	2	2	1	2	X	X	2	2	2	2	X	3	2	X	X
Sólidos inflamables (entre los que se incluyan sustancias que reaccionan espontáneamente y explosivos sólidos insensibilizados) 4.1	4	3	2	1	X	X	X	X	1	X	1	2	X	3	2	1	X
Sustancias que pueden experimentar combustión espontánea 4.2	4	3	2	2	1	2	2	1	X	1	2	2	1	3	2	1	X
Sustancias que, en contacto con el agua, desprenden gases inflamables 4.3	4	4	2	2	X	X	2	X	1	X	2	2	X	2	2	1	X
Sustancias (agentes) comburentes 5.1	4	4	2	2	X	X	2	1	2	2	X	2	1	3	1	2	X
Peróxidos orgánicos 5.2	4	4	2	2	1	2	2	2	2	2	2	X	1	3	2	2	X
Sustancias tóxicas 6.1	2	2	X	X	X	X	X	X	1	X	1	1	X	1	X	X	X
Sustancias infecciosas 6.2	4	4	4	4	2	2	3	3	3	3	3	3	1	X	3	3	X
Materiales radiactivos 7	2	2	2	2	1	1	2	2	2	2	1	2	X	3	X	2	X
Sustancias corrosivas 8	4	2	2	1	X	X	X	1	1	1	2	2	X	3	2	X	X
Sustancias y objetos peligrosos varios 9	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Ilustración 2: Cuadro de segregación. FUENTE: Código IMDG:

```

1 public String segregacion(Mmpp conENTRA, Mmpp conDETECTADO) {
2     String res = " ";
3     String CLASEconENTRA = conENTRA.getClase_3();
4     String CLASEconDETECTADO = conDETECTADO.getClase_3();
5     if (CLASEconENTRA.contains("1.1") || CLASEconENTRA.contains("1.2")
6 || CLASEconENTRA.contains("1.5")) {
7         if (CLASEconDETECTADO.contains("1.1") ||
8 CLASEconDETECTADO.contains("1.2") || CLASEconDETECTADO.contains("1.5")) {
9             res = charCompatibilidad(CLASEconENTRA, CLASEconDETECTADO);
10        } else if (CLASEconDETECTADO.contains("1.3") ||
11 CLASEconDETECTADO.contains("1.6")) {
12            res = charCompatibilidad(CLASEconENTRA, CLASEconDETECTADO);
13        } else if (CLASEconDETECTADO.contains("1.4")) {
14            res = charCompatibilidad(CLASEconENTRA, CLASEconDETECTADO);
15        } else if (CLASEconDETECTADO.equals("2.1")) {
16            res = "4";
17        } else if (CLASEconDETECTADO.equals("2.2")) {
18            res = "2";
19        } else if (CLASEconDETECTADO.equals("2.3")) {
20            res = "2";
21        } else if (CLASEconDETECTADO.equals("3")) {
22            res = "4";
23        } else if (CLASEconDETECTADO.equals("4.1")) {
24            res = "4";
25        } else if (CLASEconDETECTADO.equals("4.2")) {
26            res = "4";
27        } else if (CLASEconDETECTADO.equals("4.3")) {
28            res = "4";
29        } else if (CLASEconDETECTADO.equals("5.1")) {
30            res = "4";

```

```

31         } else if (CLASEconDETECTADO.equals("5.2")) {
32             res = "4";
33         } else if (CLASEconDETECTADO.equals("6.1")) {
34             res = "2";
35         } else if (CLASEconDETECTADO.equals("6.2")) {
36             res = "4";
37         } else if (CLASEconDETECTADO.equals("7")) {
38             res = "2";
39         } else if (CLASEconDETECTADO.equals("8")) {
40             res = "4";
41         } else if (CLASEconDETECTADO.equals("9")) {
42             res = "X";
43         }
44     } else if (CLASEconENTRA.contains("1.3") ||
45 CLASEconENTRA.contains("1.6")) {
46         if (CLASEconDETECTADO.contains("1.1") ||
47 CLASEconDETECTADO.contains("1.2") || CLASEconDETECTADO.contains("1.5")) {
48             res = charCompatibilidad(CLASEconENTRA, CLASEconDETECTADO);
49         } else if (CLASEconDETECTADO.contains("1.3") ||
50 CLASEconDETECTADO.contains("1.6")) {
51             res = charCompatibilidad(CLASEconENTRA, CLASEconDETECTADO);
52         }
53     }
54     Continúa 400 líneas...

```

Siendo necesario para las clases de explosivos un método específico para su estiba mixta, `charCompatibilidad`, que introduciendo los grupos de compatibilidad `c0` y `c1` y dos clases de explosivos, devolverá la posibilidad de que se pueda estibar de forma combinada (" ") o no ("NO").

Grupo de compatibilidad	A	B	C	D	E	F	G	H	J	K	L	N	S
A	X												
B		X											X
C			X	X ⁵	X ⁵		X ¹					X ⁴	X
D			X ⁵	X	X ⁵		X ¹					X ⁴	X
E			X ⁵	X ⁵	X		X ¹					X ⁴	X
F						X							X
G			X ¹	X ¹	X ¹		X						X
H								X					X
J									X				X
K										X			X
L											X ²		
N			X ⁴	X ⁴	X ⁴							X ³	X ⁵
S		X	X	X	X	X	X	X	X	X		X ⁵	X

La «X» indica que las mercancías de los correspondientes grupos de compatibilidad pueden estibarse en el mismo compartimiento, bodega o unidad de transporte cerrada.

Ilustración 3: Cuadro para la estiba mixta. FUENTE: Código IMDG.

```

1 public String charCompatibilidad(String c0, String c1) {
2     String res = "NO";
3     if (grupoCompatibilidad(c0) == 'A') {
4         if (grupoCompatibilidad(c1) == 'A') {
5             res = " ";
6         }
7     }
8     if (grupoCompatibilidad(c0) == 'B') {
9         if (grupoCompatibilidad(c1) == 'B' || grupoCompatibilidad(c1) ==
10 'S') {

```

```

11         res = " ";
12     }
13 }
14     if (grupoCompatibilidad(c0) == 'C') {
15         if (grupoCompatibilidad(c1) == 'C' || grupoCompatibilidad(c1) ==
16 'D' || grupoCompatibilidad(c1) == 'E' || grupoCompatibilidad(c1) == 'G' ||
17 grupoCompatibilidad(c1) == 'N' || grupoCompatibilidad(c1) == 'S') {
18             res = " ";
19         }
20     }
21     if (grupoCompatibilidad(c0) == 'D') {
22         if (grupoCompatibilidad(c1) == 'C' || grupoCompatibilidad(c1) ==
23 'D' || grupoCompatibilidad(c1) == 'E' || grupoCompatibilidad(c1) == 'G' ||
24 grupoCompatibilidad(c1) == 'N' || grupoCompatibilidad(c1) == 'S') {
25             res = " ";
26         }
27     }
28     if (grupoCompatibilidad(c0) == 'E') {
29         if (grupoCompatibilidad(c1) == 'C' || grupoCompatibilidad(c1) ==
30 'D' || grupoCompatibilidad(c1) == 'E' || grupoCompatibilidad(c1) == 'G' ||
31 grupoCompatibilidad(c1) == 'N' || grupoCompatibilidad(c1) == 'S') {
32             res = " ";
33         }
34     }
35     if (grupoCompatibilidad(c0) == 'F') {
36         if (grupoCompatibilidad(c1) == 'F' || grupoCompatibilidad(c1) ==
37 'S') {
38             res = " ";
39         }
40     }
41     if (grupoCompatibilidad(c0) == 'G') {
42         if (grupoCompatibilidad(c1) == 'C' || grupoCompatibilidad(c1) ==
43 'D' || grupoCompatibilidad(c1) == 'E' || grupoCompatibilidad(c1) == 'G' ||
44 grupoCompatibilidad(c1) == 'S') {
45             res = " ";
46         }
47     }
48     if (grupoCompatibilidad(c0) == 'H') {
49         if (grupoCompatibilidad(c1) == 'H' || grupoCompatibilidad(c1) ==
50 'S') {
51             res = " ";
52         }
53     }
54     if (grupoCompatibilidad(c0) == 'J') {
55         if (grupoCompatibilidad(c1) == 'J' || grupoCompatibilidad(c1) ==
56 'S') {
57             res = " ";
58         }
59     }
60     if (grupoCompatibilidad(c0) == 'K') {
61         if (grupoCompatibilidad(c1) == 'K' || grupoCompatibilidad(c1) ==
62 'S') {
63             res = " ";
64         }
65     }
66     if (grupoCompatibilidad(c0) == 'L') {
67         if (grupoCompatibilidad(c1) == 'L' || grupoCompatibilidad(c1) ==

```

```

68 'S') {
69     res = " ";
    }
}
    if (grupoCompatibilidad(c0) == 'N') {
        if (grupoCompatibilidad(c1) == 'B' || grupoCompatibilidad(c1) ==
'C' || grupoCompatibilidad(c1) == 'D' || grupoCompatibilidad(c1) == 'E' ||
grupoCompatibilidad(c1) == 'G' || grupoCompatibilidad(c1) == 'N' ||
grupoCompatibilidad(c1) == 'S') {
            res = " ";
        }
    }
    if (grupoCompatibilidad(c0) == 'S') {
        if (grupoCompatibilidad(c1) == 'B' || grupoCompatibilidad(c1) ==
'C' || grupoCompatibilidad(c1) == 'D' || grupoCompatibilidad(c1) == 'E' ||
grupoCompatibilidad(c1) == 'F' || grupoCompatibilidad(c1) == 'G' ||
grupoCompatibilidad(c1) == 'H' || grupoCompatibilidad(c1) == 'J' ||
grupoCompatibilidad(c1) == 'K' || grupoCompatibilidad(c1) == 'N' ||
grupoCompatibilidad(c1) == 'S') {
            res = " ";
        }
    }
    return res;
}

```

4.1.5. CLASE ESCOTILLA

Uso.

La clase escotilla sirve para realizar una interacción realista entre los conjuntos de contenedores de cubierta y de bodega.

- Este comprobante se utiliza a partir de la iteración 4 del algoritmo de carga de la clase **Viaje**. Se verá modificado conforme se vaya estibando el barco. Vendría a ser un “interruptor” que permite el paso de contenedores dependiendo del estatus cubierta-bodega.

Esta clase importa la librería de Oracle **Serializable**, necesaria para el guardado de la información.

Declaración de atributos.

```
1 boolean llenarCubierta;  
2 int reserva;  
3 int teuTotales;  
4 int teuOcupados;
```

- llenarCubierta (**boolean**): Este valor es detectado por el algoritmo de carga para saber si puede llenar la cubierta, se inicializa como false.
- reserva (**int**): Indica la reserva que sale antes de todo el conjunto de contenedores de la bodega.
- teuTotales (**int**): Indica el los TEUs que tiene la bodega. Constante.
- teuOcupados (**int**): Indica los TEUs que ocupados en bodega.

Constructor.

```
1 Escotilla() {}  
2  
3 public Escotilla(boolean llenarCubierta, int reserva, int teuTotales, int  
   teuOcupados) {  
4     this.llenarCubierta = llenarCubierta;  
5     this.reserva = reserva;  
6     this.teuTotales = teuTotales;  
7     this.teuOcupados = teuOcupados;  
8 }
```

4.1.6. CLASE CONTENEDOR

Uso.

Esta clase es fundamental para la comprensión del programa y unifica tres de los objetos vistos anteriormente. Los datos tienen en consideración la norma UNE-EN ISO 6346 y sus métodos derivan de ésta.

Esta clase se utilizará para:

- En la clase **Barco** para la elaboración de los planos de carga.
- En la clase **Ruta** para los packing lists.

Esta clase importa las siguientes librerías de Oracle:

```
1 import java.io.EOFException;
2 import java.io.FileInputStream;
3 import java.io.FileOutputStream;
4 import java.io.IOException;
5 import java.io.ObjectInputStream;
6 import java.io.ObjectOutputStream;
7 import java.io.Serializable;
```

Declaración de atributos.

```
1 String id;
2 String dyt;
3 Puerto pol;
4 Puerto pod;
5 double pesoTara;
6 double pesoUtil;
7 double pesoCarga;
8 double vgm;
9 Mmpp mmpp;
10 int listado;
11 int reserva;
12 Empresa empresa;
```

- **id (String):** eh Sistema de identificación Compuesto por:
 - Código del propietario de tres letras mayúsculas (se obtiene a partir del atributo bic de empresa). Es único y está registrado en la Oficina Internacional de Contenedores, o un organismo nacional afiliado a éste.
 - Identificador de categoría de equipo. Consta de una letra mayúscula. Como siempre tratamos con contenedores,

siempre será la U

- Número de serie de seis cifras árabes. Si son menos de seis se añadirán ceros hasta obtener un total de seis cifras.
- Una cifra de autocontrol para la verificación de números de serie y propiedad. Obtenida en el método

`determinarCifraAutocontrol`.

- `dvt (String)`: Código de dimensiones y de tipo de contenedor.

Compuesto por cuatro caracteres:

- Los dos primeros son para las dimensiones:
 - Primer carácter: Alfanumérico que representa la longitud.
 - Segundo carácter: Alfanumérico que representa la anchura y la altura.
- Las dos últimas para el tipo:
 - Tercer carácter: Alfabético que representa el tipo del contenedor.
 - Cuarto carácter: Numérico que representa las características relacionadas con el tipo.

- `pol (Puerto)`: Puerto de origen (Port of Loading).
- `pod (Puerto)`: Puerto de destino (Port of Discharge).
- `pesoTara (double)`: Peso del contenedor vacío, se obtiene por medio de su `dvt`.
- `pesoUtil (double)`: Peso máximo de la carga posible del contenedor, se obtiene por medio de su `dvt`.
- `pesoCarga (double)`: Peso de la carga del contenedor, incluye sus elementos de estiba.
- `vgm (double)`: Masa bruta del contenedor. Se obtiene por medio del método `calcularVGM`.
- `mmpp (Mmpp)`: Indicador de si el contenedor lleva una mercancía peligrosa, de llevar, mostrará la información sobre esta y como se ha de cargar.
- `listado (int)`: Número de listado de contenedores. Indica en que posición se encuentra el packing list del contenedor con respecto a la

ruta que tiene que hacer.

- reserva (`int`): Número de reserva. Indica la posición del contenedor dentro del packing list.
- empresa (`Empresa`): Empresa a la que pertenece el contenedor.

Constructor.

```
1 public Contenedor(int listado, int reserva, String id, String dyt, Puerto
  pol, Puerto pod, double pesoTara, double pesoUtil, double pesoCarga, double
  vgm, Mmpp mmpp, Empresa empresa) {
2     this.listado = listado;
3     this.reserva = reserva;
4     this.id = id;
5     this.dyt = dyt;
6     this.pol = pol;
7     this.pod = pod;
8     this.pesoTara = pesoTara;
9     this.pesoUtil = pesoUtil;
10    this.pesoCarga = pesoCarga;
11    this.vgm = vgm;
12    this.mmpp = mmpp;
13    this.empresa = empresa;
14 }
```

Nota: El cálculo del VGM no está implementado dentro del constructor porque hay veces en los que el objeto contenedor no se va a utilizar como si fuese un contenedor real. Para más información, consultar el apartado 5.3. de este documento.

Métodos secundarios de interés.

Esta función sustituye los tres primeros dígitos de una id de contenedor por los de la empresa a la que pertenece.

```
1 public void bicId() {
2     char[] conv = this.empresa.getBic().toCharArray();
3     char[] conv2 = this.id.toCharArray();
4     conv2[0] = conv[0];
5     conv2[1] = conv[1];
6     conv2[2] = conv[2];
7     setId(String.valueOf(conv2));
8 }
```

Este método es para el cálculo del dígito de autocontrol de un contenedor.

```
1 public void determinarCifraAutocontrol() {
2     int nAutocontrol = 0;
3     for (int i = 0; i < 10; i++) {
4         if (i <= 3) {
```

```

5         if ('A' == this.id.charAt(i)) {
6             nAutocontrol = (int) (Math.pow(2, i) *
              (this.id.charAt(i) - 55)) + nAutocontrol;
7         } else if ('B' == this.id.charAt(i) || 'C' ==
              this.id.charAt(i) || 'D' == this.id.charAt(i) || 'E' == this.id.charAt(i)
              || 'F' == this.id.charAt(i) || 'G' == this.id.charAt(i) || 'H' ==
              this.id.charAt(i) || 'I' == this.id.charAt(i) || 'J' == this.id.charAt(i)
              || 'K' == this.id.charAt(i)) {
8             nAutocontrol = (int) (Math.pow(2, i) *
              (this.id.charAt(i) - 54)) + nAutocontrol;
9         } else if ('L' == this.id.charAt(i) || 'M' ==
              this.id.charAt(i) || 'N' == this.id.charAt(i) || 'O' == this.id.charAt(i)
              || 'P' == this.id.charAt(i) || 'Q' == this.id.charAt(i) || 'R' ==
              this.id.charAt(i) || 'S' == this.id.charAt(i) || 'T' == this.id.charAt(i)
              || 'U' == this.id.charAt(i)) {
10            nAutocontrol = (int) (Math.pow(2, i) *
              (this.id.charAt(i) - 53)) + nAutocontrol;
11        } else {
12            nAutocontrol = (int) (Math.pow(2, i) *
              (this.id.charAt(i) - 52)) + nAutocontrol;
13        }
14    } else {
15        nAutocontrol = (int) (Math.pow(2, i) * (this.id.charAt(i) -
              48)) + nAutocontrol;
16    }
17    }
18    nAutocontrol = nAutocontrol % 11;
19    if (nAutocontrol == 10) {
20        nAutocontrol = 0;
21    }
22    char[] conv = this.id.toCharArray();
23    conv[10] = (char) (nAutocontrol + 48);
24    setId(String.valueOf(conv));
25 }

```

Cuando se genera un contenedor, la posición de autocontrol (la 11) está ocupada por el carácter '?'. Primero detecta todos los valores a lo largo de la cadena de texto, realiza el cálculo descrito en el anexo A de la norma UNE-EN ISO 6346 y al final convierte el resultado en un carácter que sustituirá al '?' original.

El siguiente método es del cálculo del VGM en base a un contenedor (**this**).

```

1 public void calcularVGM() {
2     this.setVgm((double) Math.round(this.pesoTara + this.pesoCarga) *
      100d / 100d);
3 }

```

El siguiente método engloba a los anteriores. Actualiza la id y asigna al azar el peso de su carga.

Esto se hace enfocado a la generación de listas.

```
1 public void actualizarContenedor() {  
2     this.bicId();  
3     this.determinarCifraAutocontrol();  
4     this.calcularPesoTaraYPesoUtil();  
5     this.setPesoCarga(Math.round(this.getPesoUtil() * Math.random() *  
    100) / 100d);  
6     this.calcularVGM();  
7 }
```

4.1.7. CLASE BARCO

Uso.

Como indica su nombre, la clase barco representa un barco virtual. Cuando se ejecute el código se crearán una serie de barcos copia que representarán cada uno de los estados en los que se encontraba la carga a lo largo de la ruta.

Esta clase importa las siguientes librerías de Oracle:

```
1 import java.io.EOFException;
2 import java.io.FileInputStream;
3 import java.io.FileOutputStream;
4 import java.io.IOException;
5 import java.io.ObjectInputStream;
6 import java.io.ObjectOutputStream;
7 import java.io.Serializable;
8 import java.util.ArrayList;
```

Declaración de atributos.

```
1 String nombre;
2 int imo;
3 int mmsi;
4 int anyoCons;
5 String bandera;
6 Empresa empresa;
7 int teu;
8 Contenedor[][][] matriz;
9 char[] tipoCon;
10 Escotilla[] escotilla;
```

- nombre (`String`): Nombre genérico del barco.
- imo (`int`): Número IMO. Identificador del barco de siete dígitos.
- mmsi (`int`): Número MMSI. Número de identificación de nueve dígitos de la estación del barco.
- anyoCons (`int`): Año de construcción.
- bandera (`String`): País donde se encuentra abanderado.
- empresa (`Empresa`): Compañía propietaria,
- teu (`int`): Teus del barco. Entendido como huecos para contenedores de 20' disponibles. Un método se encarga de contarlos.
- matriz (`Contenedor[][][]`): Matriz de contenedores, parte

fundamental del programa. Es el espacio dentro del barco donde se distribuyen los contenedores. Cuenta con cinco dimensiones en las que cada una de ellas representa una sección del plano de carga.

Al tratarse de un array, Java lo leerá contando el primer elemento de cada listado de 0 en adelante. Sin embargo, esto no se corresponde con la numeración real que siguen los planos de bahías de los portacontenedores.

Lo conveniente es primero identificar cómo funciona cada uno de estos sistemas de posicionamiento y lo segundo, crear métodos que puedan realizar conversiones entre ambos sistemas.

Sistema 1:

Posicionamiento por arrays

Es la forma en la que el programa entiende la matriz y como se ha codificado:

Contenedor[Bahía dupla][Bahía unitaria][Bodega/Cubierta][Fila][Altura]

Donde:

Cada una de las dimensiones [] representa un espacio del buque.

Sistema 2:

Posicionamiento real

Es el que entiende el usuario y el sistema que siguen los portacontenedores:

Bahía-Fila-Nivel (Bay-Row-Tier)

BBRRTT

Donde:

BB: Se corresponde con la bahía donde está el contenedor.

RR: Se corresponde con la fila donde está el contenedor.

TT: Se corresponde con la altura donde está el contenedor.

Para que quede claro cómo funciona la matriz vamos a ir viendo cada uno de sus niveles:

- **Bahía dupla ([])**: Es cada uno de los conjuntos de bahías unitarias que se distribuyen a lo largo del buque. Está compuesta por lo menos de dos bahías unitarias.
 - **Posicionamiento por arrays**: De proa a popa.
 - **Posicionamiento real**: De proa a popa se numera a partir de 2 y van de cuatro en cuatro. Se utiliza si hay que posicionar un contenedor de 40' o 45'.



Ilustración 4: Diagrama de bahías duplas (coloreadas en verde o blanco). FUENTE: Propia.

- **Bahía unitaria ([][])**: Pueden ser una, dos o ninguna y representar un hueco en el barco que podría simular la habitación del buque o la estructura de las chimeneas. Un contenedor de 20' ocupa una bahía unitaria, uno de 40'/45' ocupa dos.
 - **Posicionamiento por arrays**: De proa a popa.
 - **Posicionamiento real**: De proa a popa. El valor de la bahía dupla que la contiene +1 si es la sección de proa y si es en popa -1.

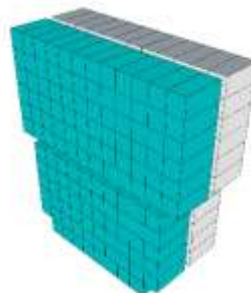


Ilustración 5: Diagrama de bahías simples (coloreadas en azul verdoso o blanco). FUENTE: Propia.

Bodega/Cubierta (□□□□): Esta dimensión representa si la posición es en cubierta o bajo la escotilla. Puede faltar una de las dos, y por norma general es la bodega, al tener que dejar espacio a la sala de máquinas.

- **Posicionamiento por arrays:** La primera posición representa la bodega, la segunda es la cubierta.
- **Posicionamiento real:** Que se sepa si un contenedor está en bodega o cubierta se deduce en base a los dígitos de nivel (tier, TT), especificados en la dimensión altura.

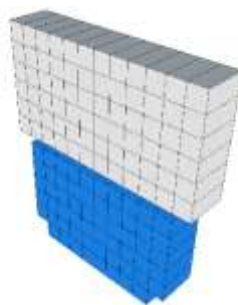


Ilustración 6: Diagrama de cubierta/bodega (coloreadas en azul o blanco). FUENTE: Propia.

Fila (□□□□□): Correspondientes a la sección transversal de contenedores apilados.

- **Posicionamiento por arrays:** Se cuenta de babor a estribor.
- **Posicionamiento real:** Se cuenta de babor a estribor. Se numerarán de la crujía hacia los costados. En el lado de babor se numeran de dos en dos partiendo de 2. Las de estribor partiendo de 1 de dos en dos. Por lo que en babor quedan numeradas con pares y en estribor impares. Si el número de filas es par, y por lo tanto hay columna central, se numerará como 0.

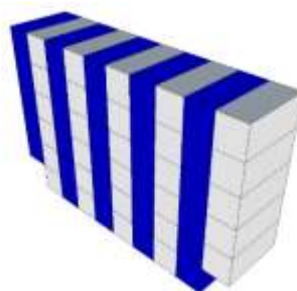


Ilustración 7: Diagrama de filas (coloreadas en azul oscuro o blanco). FUENTE: Propia.

Altura (000000): Posición del contenedor dentro su columna.

- **Posicionamiento por arrays:** Se enumeran de abajo a arriba.
- **Posicionamiento real:** Se enumeran de abajo a arriba. Los contenedores en bodega se numeran a partir de 2, los de cubierta a partir de 82. Por cada nivel que ascienden se suma dos.



Ilustración 8: Diagrama de alturas (coloreadas en rojo o blanco). FUENTE: Propia.

Con el fin de que esta parte quede clara vamos a ver el siguiente ejemplo, en el cual vamos a identificar los dos sistemas de posicionamiento de tres contenedores.

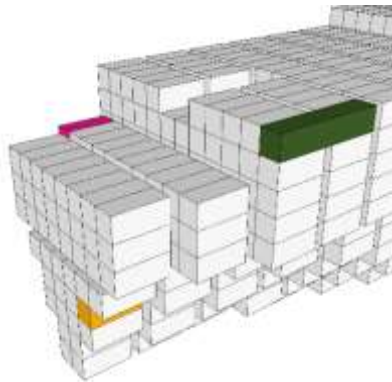


Ilustración 9: Proa de un portacontenedores. Véanse tres contenedores coloreados. FUENTE: Propia.

Las posiciones de los contenedores coloreados serán las siguientes:

- El contenedor naranja de 20'. Está localizado en la primera bahía dupla, primera bahía unitaria, en bodega, en la fila segunda contando desde babor, segundo nivel:
 - Posicionamiento por array:
`Contenedor[0][0][0][1][1]`
 - Posicionamiento real:
`000408`
- El contenedor magenta de 20'. Está en la primera bahía dupla,

segunda bahía unitaria, en cubierta, primera fila desde estribor, tercera altura:

- Posicionamiento por array:

`Contenedor[0][1][1][9][2]`

- Posicionamiento real:

010986

- El contenedor verde oscuro, simula ser un 40', por lo tanto, ocupa dos espacios de 20', en cubierta, primera fila de babor, quinta altura:

- Posicionamiento por array. El 40' estará en la bahía simple de proa y en la de popa. Se creará un contenedor "copia" que simula el segundo espacio que ocupa:

`Contenedor[1][0][1][0][4]`

(para el contenedor)

`Contenedor[1][1][1][0][4]`

(para la copia que simula ser la prolongación del 40')

- Posicionamiento real. Al ocupar las dos bahías simples (5 y 7) tendrá el valor de su bahía dupla (6):

061290

- tipoCon (`char[]`): Array que indica el tipo de contenedores que se pueden introducir en la bahía dupla. Tiene la misma longitud que de bahías duplas y cada una de sus posiciones se corresponde con la de la bahía dupla que representa. Los caracteres reservados para este array son:
 - 'X': Bahía no utilizada por que en su hueco se encuentra la habilitación del barco.
 - 'a': Bahía para contenedores de 20'.
 - 'b': Bahía para contenedores de 20' y 40'.
 - 'c': Bahía para contenedores de 40'.
 - 'd': Bahía para contenedores de 40' y 45'.
 - 'e': Bahía para contenedores de 20', 40' y 45'.
- escotilla (`Escotilla[]`): Array que representa las escotillas que separan las secciones de bodega y cubierta. En los métodos de estiba perfeccionados se utiliza como un comprobante para ver si se puede cargar la parte superior del barco. Al igual que tipoCon tiene la misma

longitud que de bahías duplas.

Constructor.

```
1 public Barco(String nombre, int imo, int mmsi, int anyoCons, String bandera,
  Empresa empresa, int teu, Contenedor[][][] matriz, char[] tipoCon,
  Escotilla[] escotilla) {
2     this.nombre = nombre;
3     this.imo = imo;
4     this.mmsi = mmsi;
5     this.anyoCons = anyoCons;
6     this.bandera = bandera;
7     this.empresa = empresa;
8     this.teu = teu;
9     this.matriz = matriz;
10    this.tipoCon = tipoCon;
11    this.escotilla = escotilla;
12    for (int i = 0; i < matriz.length; i++) {
13        this.escotilla[i].setTeuTotales(contarTEUsBodega(i));
14        System.out.println(this.escotilla[i].getTeuTotales());
15    }
16 }
```

Métodos secundarios de interés.

La clase barco contiene muchos métodos secundarios, algunos reseñables son los siguientes:

Este es el método que contabiliza los TEUs que tiene un barco.

```
1 public int contarTEUs() {
2     int res = 0;
3     for (int i = 0; i < this.matriz.length; i++) {
4         for (int j = 0; j < this.matriz[i].length; j++) {
5             if (this.matriz[i][j] != null) {
6                 for (int k = 0; k < this.matriz[i][j].length; k++) {
7                     if (this.matriz[i][j][k] != null) {
8                         for (int m = 0; m < this.matriz[i][j][k].length;
9 m++) {
10                             for (int n = 0; n <
11 this.matriz[i][j][k][m].length; n++) {
12                                 if
13 (!matriz[i][j][k][m][n].getId().equals("ESTRUCTURAL")) {
14                                     res++;
15                                 }
16                             }
17                         }
18                     }
19                 }
20             }
21 }
```

Consta de una serie de bucles `for` que van sumando todas las posiciones (sumando en `res` los contenedores que hay por cada fila).

Este método es similar pero solo cuenta los espacios libres:

```
1 public int contarEspacioDisponible() {
2     int res = 0;
3     for (int i = 0; i < this.matriz.length; i++) {
4         for (int j = 0; j < this.matriz[i].length; j++) {
5             if (this.matriz[i][j] != null) {
6                 for (int k = 0; k < this.matriz[i][j].length; k++) {
7                     if (this.matriz[i][j][k] != null) {
8                         for (int m = 0; m < this.matriz[i][j][k].length;
9 m++) {
10                             for (int n = 0; n <
11 this.matriz[i][j][k][m].length; n++) {
12                                 if
13 (matriz[i][j][k][m][n].getId().equals("HUECO")) {
14                                     res++;
15                                 }
16                             }
17                         }
18                     }
19                 }
20             }
21         }
22     }
23     return res;
24 }
```

El cual si detecta que la posición no está ocupada sumará un valor al contador (`res`). Hay varios métodos que se dedican a averiguar las dimensiones máximas del barco, esto es necesario para poder dibujarlo en el visor, ya que el programa tiene que saber de antemano como distribuir los espacios en el dibujo, para esto, calculará la altura máxima, anchura máxima... Etcétera.

Para el caso del número de filas máximo (anchura), se usa este método:

```
1 public int filaMasLargaDelBarco() {
2     int res = 0;
3     for (int i = 0; i < this.matriz.length; i++) {
4         for (int j = 0; j < this.matriz[i].length; j++) {
5             if (this.matriz[i][j] != null) {
6                 for (int k = 0; k < this.matriz[i][j].length; k++) {
7                     if (this.matriz[i][j][k] != null && res <
8 matriz[i][j][k].length) {
9                         res = matriz[i][j][k].length;
10                     }
11                 }
12             }
13         }
14     }
15     return res;
16 }
```

Otros métodos se encargan de la conversión entre el sistema de coordenadas de array y real.

Este es para convertir entre un nivel en coordenada real a coordenada array.

```
1 public int conversion_posnivelAposarray(int conv, int esBodega, int longFila)
2 {
3     if (esBodega == 0) {
4         return (-conv + longFila) / 2;
5     } else {
6         return (conv - 82) / 2;
7     }
8 }
```

En el cual `conv` es el dato para convertir, `esBodega` es un indicador para saber si el contenedor se encuentra en la bodega (0) o en cubierta (1) y la `longFila` la longitud del nivel en el que se encuentra.

Este método es para obtener un contenedor dada una bahía, fila y nivel.

```
1 public Contenedor verContenedor(int bahia, int fila, int nivel) {
2     try {
3         int convBAH = 0;
4         if ((bahia + 1 % 4) == 0) {
5             convBAH = 1;
6         }
7         int convBOD = 0;
8         if (nivel >= 82) {
9             convBOD = 1;
10        }
11        int convFILA = conversion_posfilaAposarray(fila,
12        this.matriz[conversion_posbahiaAposarray(bahia)][convBAH][convBOD].length);
13        int convNIVEL = conversion_posnivelAposarray(nivel, convBOD,
14        this.matriz[conversion_posbahiaAposarray(bahia)][convBAH][convBOD][convFILA]
15        ].length);
16        return
17        this.matriz[conversion_posbahiaAposarray(bahia)][convBAH][convBOD][convFILA]
18        ][convNIVEL];
19    } catch (Exception e) {
20        System.out.println(ANSI_RED + "Posición " +
21        String.format("%02d", bahia) + String.format("%02d", fila) +
22        String.format("%02d", nivel) + " inexistente." + ANSI_RESET);
23        return null;
24    }
25 }
```

Si bien la forma preferible y que se va a utilizar en el programa es por medio de la lectura por datos de array.

Por último, nos encontramos con métodos que se encargan de obtener diversas estadísticas. Por ejemplo, este método obtiene el sumatorio de los pesos de contenedores de una columna:

```
1 public double toneladasColumna(int bahiaDupla, int bahia, int esBodega, int
  fila) {
2     double res = 0;
3     for (int i = 0; i <
  this.matriz[bahiaDupla][bahia][esBodega][fila].length; i++) {
4         if
  (this.matriz[bahiaDupla][bahia][esBodega][fila][i].longitudContenedor() <=
  6.068) {
5             res = res +
6 this.matriz[bahiaDupla][bahia][esBodega][fila][i].getVgm();
7         } else {
8             res = res +
9 this.matriz[bahiaDupla][bahia][esBodega][fila][i].getVgm() / 2;
10        }
11    }
12    return res;
13 }
```

4.1.8. CLASE RUTA

Uso.

La clase ruta indica que recorrido va a efectuar el **Barco** y los packing lists asociados a cada una de las escalas. En esta clase se aúnan objetos vistos anteriormente **Puerto**, **Contenedor** y **Fecha**.

Esta clase importa las siguientes librerías de Oracle:

```
1 import java.io.EOFException;
2 import java.io.FileInputStream;
3 import java.io.FileOutputStream;
4 import java.io.IOException;
5 import java.io.ObjectInputStream;
6 import java.io.ObjectOutputStream;
7 import java.io.Serializable;
8 import java.util.ArrayList;
```

Declaración de atributos.

```
1 String nombre;
2 Contenedor[] cargaPrevia;
3 ArrayList<Puerto> puertos;
4 ArrayList<Contenedor[]> listaDeListas;
5 ArrayList<Fecha> tiempo;
```

- nombre (String): Nombre de la ruta.
- cargaPrevia (Contenedor[]): Packing list de contenedores que se encontraban en el barco antes de haber iniciado la ruta.
- puertos (ArrayList<Puerto>): Listado de puertos que se van a recorrer. Este orden es fundamental para los algoritmos de estiba puesto que establecerá la secuencia con la que van a entrar los packing list.
- listaDeListas (ArrayList<Contenedor[]>): Como indica su nombre, es una lista de listados de contenedores. Cada uno de los listados se corresponde con el elemento de su mismo índice del arrayList puertos.
- tiempo (ArrayList<Fecha>): Tiempos invertidos entre cada una de las visitas. En su primer elemento se corresponde con el tiempo que tardará el buque en llegar desde su posición inicial desconocida al primer puerto.

Constructor.

```
1 public Ruta() {}
2
3 public Ruta(String nombre, Contenedor[] cargaPrevia, ArrayList<Puerto>
  puertos, ArrayList<Contenedor[]> listaDeListas, ArrayList<Fecha> tiempo) {
4     this.nombre = nombre;
5     this.cargaPrevia = cargaPrevia;
6     this.puertos = puertos;
7     this.listaDeListas = listaDeListas;
8     this.tiempo = tiempo;
9 }
```

Métodos secundarios de interés.

Este método, `toStringInforme`, imprime por consola los packing lists que se introducen en cada puerto.

```
1 public String toStringInforme() {
2     Contenedor con = new Contenedor();
3     Puerto pue = new Puerto();
4     String s = this.nombre;
5     s = s + "\n\n";
6     if (this.cargaPrevia != null) {
7         s = s + "Con carga previa\n";
8         for (int i = 0; i < this.cargaPrevia.length; i++) {
9             s = s + cargaPrevia[i].toStringClase("lis", "lisM", "lis") +
10             "\n";
11         }
12     } else {
13         s = s + "Sin carga previa\n";
14     }
15     s = s + "\n\n";
16     s = s + "Con la siguiente ruta:";
17     s = s + "\n";
18     for (int j = 0; j < this.listaDeListas.size(); j++) {
19         s = s + "\n";
20         s = s + (j + 1) + "° carga en el puerto de " +
21         this.puertos.get(j + 1);
22         s = s + "\n";
23         s = s + "Al que se llegó tras un viaje de " +
24         this.tiempo.get(j);
25         s = s + "\n";
26         for (int i = 0; i < this.listaDeListas.get(j).length; i++) {
27             s = s + listaDeListas.get(j)[i].toStringClase("lis", "lisM",
28             "lis") + "\n";
29         }
30     }
31     return s;
32 }
```

4.1.9. CLASE VIAJE

Uso.

La clase viaje es la más compleja del paquete clases y unifica todo lo enseñado hasta el momento. Sus objetos serán el paquete de información que contendrá los inputs y que el algoritmo de carga utilizará.

Lo más relevante y en lo que quiero hacer énfasis, es que aquí hacemos la interacción entre el “vehículo” **Barco** y el “trazado” que es **Ruta**.

Esta clase importa las siguientes librerías de Oracle:

```
1 import java.io.EOFException;
2 import java.io.FileInputStream;
3 import java.io.FileOutputStream;
4 import java.io.IOException;
5 import java.io.ObjectInputStream;
6 import java.io.ObjectOutputStream;
7 import java.io.Serializable;
8 import java.util.ArrayList;
```

Declaración de atributos.

```
1 String nombre;
2 Barco barco;
3 Barco[] cargas;
4 Ruta ruta;
5 int version;
```

- nombre (`String`): Nombre del viaje.
- barco (`Barco`): Barco que va a efectuar el recorrido.
- cargas (`Barco[]`): Cada una de las cargas en las que el barco va a efectuar. Las funciones de estiba harán una copia del barco según está cargado en la salida de puerto y la alojarán aquí. Su longitud se corresponde al número de cargas a realizar más una correspondiente a la situación inicial.
- ruta (`Ruta`): Recorrido que va a efectuar el barco.
- versión (`int`): Número que indica la versión de algoritmo que va a utilizarse en la carga.

Constructor.

```
1 public Viaje(Barco barco, Ruta ruta, int version) {
2     this.nombre = barco.getNombre() + " en el trayecto " +
    ruta.getNombre();
3     this.barco = barco;
4     this.ruta = ruta;
5     this.version = version;
6     if (version == 0) {
7         this.cargas = this.cargaBasica_v0();
8     } else if (version == 1) {
9         this.cargas = this.cargaBasica_v1();
10    } else if (version == 2) {
11        this.cargas = this.cargaBasica_v2();
12    } else if (version == 3) {
13        this.cargas = this.cargaBasica_v3();
14    } else if (version == 4) {
15        this.cargas = this.cargaBasica_v4();
16    } else if (version == 5) {
17        this.cargas = this.cargaBasica_v5();
18    } else {
19        this.cargas = this.cargaBasica_v0();
20    }
21 }
```

Métodos secundarios de interés.

En la clase viaje se contienen los métodos que se encargan de cargar los barcos, debido a la extensión de estos y a su importancia, tienen un apartado dedicado a estos en el capítulo 5 de este trabajo.

4.2. PAQUETE IMDG

En este paquete se encuentra una serie de clases en las que se ha tabulado el código IMDG. Esto se hace en base al objeto creado en la clase **Mmpp** visto en el capítulo 4.1.4.

Su finalidad es que cuando se generen las listas de contenedores, existe una pequeña probabilidad de que se genere mercancía IMO, el generador buscará en estas listas y escogerá un número ONU al azar.

Nota: Si un contenedor carece de mercancía IMO, en su atributo **Mmpp** lo contendrá un objeto especial que representa esa ausencia:

```
1 final protected String[] ARNO = {"-"};
2 final public Mmpp MMPP_0 = new Mmpp(0, "", "", ARNO, "", ARNO, ARNO);
```

El paquete contiene las siguientes clases:

- **ImdgCategorías:** Donde se almacenan más de doscientas variedades de riesgo secundario (4) y los tipos de criterios de estiba y segregación (16a, 16b). Siendo un array de dos posiciones en el caso del riesgo secundario y para los otros dos de longitud variable. Los fragmentos de texto que alojan son interpretados por las funciones de la clase **Mmpp** y en base a éstos se tomarían decisiones de cara a la segregación.

```
1 //Ejemplos para el riesgo
2 final protected String[] ARri_61P = {"6.1", "P"};
3 final protected String[] ARri_78 = {"7/8", ""};
4 final protected String[] ARri_8 = {"8", ""};
5 final protected String[] ARri_8P = {"8", "P"};
6 final protected String[] ARri_P = {"", "P"};
7 final protected String[] ARri_SP172 = {"Véase SP172", ""};
8 //Ejemplos para la categoría 16a
9 final protected String[] ARse_SG20 = {"SG20"};
10 final protected String[] ARse_SG20_SG21 = {"SG20", "SG21"};
11 final protected String[] ARse_SG22_SG35 = {"SG22", "SG35"};
12 final protected String[] ARse_SG23 = {"SG23"};
13 //Ejemplos para la categoría 16b
14 final protected String[] ca_sw12 = {"Categoría A", "SW12"};
15 final protected String[] ca_sw12_sw20_sw21 = {"Categoría A", "SW12", "SW20",
16 "SW21"};
17 final protected String[] ca_sw12_sw21 = {"Categoría A", "SW12", "SW21"};
18 final protected String[] ca_sw13 = {"Categoría A", "SW13"};
19 final protected String[] ca_sw16 = {"Categoría A", "SW16"};
```

```

19 final protected String[] ca_sw19 = {"Categoría A", "SW19"};
20 final protected String[] ca_sw2_h4 = {"Categoría A", "SW2", "H4"};

```

- Clases con los elementos del código IMDG, no está todo dentro del mismo fichero de texto para evitar problemas de memoria (error java.lang.NoClassDefFoundError):

- **Imdg1a510**
- **Imdg1001a1500**
- **Imdg1502a2000**
- **Imdg2001a2498**
- **Imdg2501a2998**
- **Imdg3005a3534**

```

1 final public Mmpp MMPP_2931 = new Mmpp(2931, "SULFATO DE VANADILO", "6.1",
  imdg.ARNO, "II", imdg.ca, imdg.ARNO);
2 final public Mmpp MMPP_2933 = new Mmpp(2933, "2-CLOROPROPIONATO DE METILO",
  "3", imdg.ARNO, "III", imdg.ca, imdg.ARNO);
3 final public Mmpp MMPP_2934 = new Mmpp(2934, "2-CLOROPROPIONATO DE
  ISOPROPILO", "3", imdg.ARNO, "III", imdg.ca, imdg.ARNO);
4 final public Mmpp MMPP_2935 = new Mmpp(2935, "2-CLOROPROPIONATO DE ETILO",
  "3", imdg.ARNO, "III", imdg.ca, imdg.ARNO);
5 final public Mmpp MMPP_2936 = new Mmpp(2936, "ÁCIDO TIOLÁCTICO", "6.1",
  imdg.ARNO, "II", imdg.ca, imdg.ARNO);
6 final public Mmpp MMPP_2937 = new Mmpp(2937, "ALCOHOL alfaMETILBENCÍLICO
  LÍQUIDO", "6.1", imdg.ARNO, "III", imdg.ca, imdg.ARNO);
7 final public Mmpp MMPP_2940 = new Mmpp(2940, "9-FOSFABICICLONANOS
  (FOSFINAS DE CICLOOCTADIENO)", "4.2", imdg.ARNO, "II", imdg.ca, imdg.ARNO);
8 final public Mmpp MMPP_2941 = new Mmpp(2941, "FLUOROANILINAS", "5.1",
  imdg.ARNO, "III", imdg.ca, imdg.ARNO);
9 final public Mmpp MMPP_2942 = new Mmpp(2942, "2-TRIFLUOROMETILANILINA",
  "6.1", imdg.ARNO, "III", imdg.ca, imdg.ARNO);
10 final public Mmpp MMPP_2943 = new Mmpp(2943, "TETRAHIDROFURFURILAMINA", "3",
  imdg.ARNO, "III", imdg.ca, imdg.ARNO);
11 final public Mmpp MMPP_2945 = new Mmpp(2945, "N-METILBUTILAMINA", "3",
  imdg.ARri_8, "II", imdg.cb_sw2, imdg.ARNO);
12 final public Mmpp MMPP_2946 = new Mmpp(2946, "2-AMINO-5-DIETILAMINOPENTANO",
  "6.1", imdg.ARNO, "III", imdg.ca, imdg.ARNO);
13 final public Mmpp MMPP_2947 = new Mmpp(2947, "CLOROACETATO DE ISOPROPILO",
  "3", imdg.ARNO, "III", imdg.ca, imdg.ARNO);
14 final public Mmpp MMPP_2948 = new Mmpp(2948, "3-TRIFLUOROMETILANILINA",
  "6.1", imdg.ARNO, "II", imdg.ca_sw2, imdg.ARNO);
15 final public Mmpp MMPP_2949 = new Mmpp(2949, "HIDROSULFURO SÓDICO HIDRATADO
  - con no menos de un 25 % de agua de cristalización", "8", imdg.ARNO, "II",
  imdg.ca, imdg.ARse_SG35);
16 final public Mmpp MMPP_2950 = new Mmpp(2950, "MAGNESIO EN GRÁNULOS
  RECUBIERTOS - en partículas de no menos de 149 micrones", "4.3", imdg.ARNO,
  "III", imdg.ca_h1, imdg.ARse_SG26_SG35);
17 final public Mmpp MMPP_2956 = new Mmpp(2956, "5-terc-BUTIL-2,4,6-TRINITRO-m-
  XILENO(ALMIZCLE-XILENO)", "4.6", imdg.ARNO, "III", imdg.cd_sw1_sw2_h2_h3,
  imdg.ARse_SG1);
18 final public Mmpp MMPP_2965 = new Mmpp(2965, "ETERATO DIMETÍLICO DE

```

```

TRIFLUORURO DE BORO", "4.3", imdg.ARri_38, "I", imdg.cd_sw2_h1,
    imdg.ARse_SG5_SG8_SG13_SG25_SG26);
19 final public Mmpp MMPP_2966 = new Mmpp(2966, "TIOGLICOL", "6.1", imdg.ARNO,
    "II", imdg.ca, imdg.ARNO);
20 final public Mmpp MMPP_2967 = new Mmpp(2967, "ÁCIDO SULFÁMICO", "8",
    imdg.ARNO, "III", imdg.ca, imdg.ARNO);
21 final public Mmpp MMPP_2968 = new Mmpp(2968, "MANEB, ESTABILIZADO o
    PREPARADO DE MANEB, ESTABILIZADO - contra el calentamiento espontáneo",
    "4.3", imdg.ARri_P, "III", imdg.cb_h1, imdg.ARse_SG26_SG29_SG35);
22 final public Mmpp MMPP_2969 = new Mmpp(2969, "SEMILLAS DE RICINO o HARINA DE
    RICINO o PULPA DE RICINO o ESCAMAS DE RICINO", "9", imdg.ARNO, "II",
    imdg.ce_sw2, imdg.ARse_SG10_SG18_SG29);

```

- **ImdgTablas:** Donde se unifican las clases vistas anteriormente en una única tabla de datos.

```

1 final public Mmpp[] LISTA_IMDG = {imgd.MMPP_0004, imgd.MMPP_0005,
    imgd.MMPP_0006, imgd.MMPP_0007, imgd.MMPP_0009, imgd.MMPP_0010,
    imgd.MMPP_0012, imgd.MMPP_0014, imgd.MMPP_0015, imgd.MMPP_0016,
    imgd.MMPP_0018, imgd.MMPP_0019, imgd.MMPP_0020, imgd.MMPP_0021,
2      imgd.MMPP_0027, imgd.MMPP_0028, imgd.MMPP_0029, imgd.MMPP_0030,
    imgd.MMPP_0033, imgd.MMPP_0034, imgd.MMPP_0035, imgd.MMPP_0037,
    imgd.MMPP_0038, imgd.MMPP_0039, imgd.MMPP_0042, imgd.MMPP_0043,
    imgd.MMPP_0044, imgd.MMPP_0048, imgd.MMPP_0049, imgd.MMPP_0050,
    imgd.MMPP_0054,
3      imgd.MMPP_0055, imgd.MMPP_0056, imgd.MMPP_0059, imgd.MMPP_0060,
    imgd.MMPP_0065, imgd.MMPP_0066, imgd.MMPP_0070, imgd.MMPP_0072,
    imgd.MMPP_0073, imgd.MMPP_0074, imgd.MMPP_0075, imgd.MMPP_0076,
    imgd.MMPP_0077, imgd.MMPP_0078, imgd.MMPP_0079, imgd.MMPP_0081,
    imgd.MMPP_0082,
4      imgd.MMPP_0083, imgd.MMPP_0084, imgd.MMPP_0092, imgd.MMPP_0093,
    imgd.MMPP_0094, imgd.MMPP_0099, imgd.MMPP_0101, imgd.MMPP_0102,
    imgd.MMPP_0103, imgd.MMPP_0104, imgd.MMPP_0105, imgd.MMPP_0106,
    imgd.MMPP_0107, imgd.MMPP_0110, imgd.MMPP_0113, imgd.MMPP_0114,
    imgd.MMPP_0118,
5      imgd.MMPP_0121, imgd.MMPP_0124, imgd.MMPP_0129, imgd.MMPP_0130,
    imgd.MMPP_0131, imgd.MMPP_0132, imgd.MMPP_0133, imgd.MMPP_0135,
    imgd.MMPP_0136, imgd.MMPP_0137, imgd.MMPP_0138, imgd.MMPP_0143,
    imgd.MMPP_0144, imgd.MMPP_0146, imgd.MMPP_0147, imgd.MMPP_0150,
    imgd.MMPP_0151,
6      imgd.MMPP_0153, imgd.MMPP_0154, imgd.MMPP_0155, imgd.MMPP_0159,
    imgd.MMPP_0160, imgd.MMPP_0161, imgd.MMPP_0167, imgd.MMPP_0168,
    imgd.MMPP_0169, imgd.MMPP_0171, imgd.MMPP_0173, imgd.MMPP_0174,
    imgd.MMPP_0180, imgd.MMPP_0181, imgd.MMPP_0182, imgd.MMPP_0186,
    imgd.MMPP_0190,
7      imgd.MMPP_0191, imgd.MMPP_0192, imgd.MMPP_0193, imgd.MMPP_0194,
    imgd.MMPP_0195, imgd.MMPP_0196, imgd.MMPP_0197, imgd.MMPP_0204,
    imgd.MMPP_0207, imgd.MMPP_0208, imgd.MMPP_0209, imgd.MMPP_0212,
    imgd.MMPP_0213... };//Continúa

```

4.3. PAQUETE ASTILLERO

En este fichero se alojan los barcos codificados, vimos como su clase se creaba en el capítulo 4.1.7. Debido a la complejidad y a que se buscaba ante todo realismo, los barcos no son generables y han de ser introducidos en el programa a mano. Dentro del programa hay implementados tres barcos reales (Beatriz B, MSC Positano y Caroline Maersk).

Como trabajamos con una matriz de contenedores hay que crear un contenedor especial que simule la disponibilidad de espacio, la nada. De esta manera los algoritmos de estiba podrán detectar localizaciones donde se pueda cargar.

```
1 final public Contenedor HUECO = new Contenedor(0, 0, "HUECO", "?", null, null, 0.0, 0.0, 0.0, 0.0, 0.0, null, null);
```

Además, la matriz no será perfecta, pues dependiendo de la sección que trabajemos contará con irregularidades: Finos de proa, huecos por la presencia de tanques de lastre o estructuras en cubierta. Por lo que ha de existir un contenedor que simule su presencia.

```
1 final public Contenedor ESTRUCTURAL = new Contenedor(0, 0, "ESTRUCTURAL", "?", null, null, 0.0, 0.0, 0.0, 0.0, 0.0, null, null);
```

Las escotillas del barco, por defecto estarán abiertas y cuando se vaya cargando el barco se irán modificando.

```
1 Escotilla esc = new Escotilla(false, 0, 0, 0);
```

Dentro del paquete astillero hay una clase principal que gestionará las clases que contienen los barcos codificados. Cuando se ejecute el programa y se introduzcan los inputs, el menú llamará a este paquete con el nombre de uno de los barcos y se devolverá el barco, de no haber un barco seleccionado, se devolverá por defecto el buque Beatriz B.

Ejemplo: Buque Beatriz B



Ilustración 10: El buque Beatriz B. FUENTE: Boluda Corporación Marítima.

Contenido en la clase de su mismo nombre, la clase Beatriz B es:

```
1 final public Barco BEATRIZB = new Barco("Beatriz B", 9448671, 209056000,  
2011, "Chipre", lis.BOL, 1267, matrizBeatrizB, tipoConBeatrizB,  
escotillaBeatrizB);
```

Donde `escotillaBeatrizB` es la cadena de escotillas:

```
1 Escotilla esc_9448671_2 = new Escotilla(false, 0, 11, 0);  
2 Escotilla esc_9448671_3 = new Escotilla(false, 0, 50, 0);  
3 Escotilla esc_9448671_4 = new Escotilla(false, 0, 70, 0);  
4 Escotilla esc_9448671_5 = new Escotilla(false, 0, 78, 0);  
5 Escotilla esc_9448671_6 = new Escotilla(false, 0, 80, 0);  
6 Escotilla esc_9448671_7 = new Escotilla(false, 0, 80, 0);  
7 Escotilla esc_9448671_8 = new Escotilla(false, 0, 78, 0);  
8 Escotilla esc_9448671_9 = new Escotilla(false, 0, 70, 0);  
9 Escotilla esc_9448671_10 = new Escotilla(false, 0, 0, 0);  
10 Escotilla[] escotillaBeatrizB = {esc_9448671_2, esc_9448671_3,  
esc_9448671_4, esc_9448671_5, esc_9448671_6, esc_9448671_7, esc_9448671_8,  
esc_9448671_9, esc_9448671_10};
```

Para denominar el tipo de contenedores que entran en bodega tenemos la cadena de caracteres `tipoConBeatrizB`:

```
1 char[] tipoConBeatrizB = {'a', 'b', 'e', 'b', 'e', 'b', 'b', 'b', 'b'};
```

Y su matriz de contenedores, cada uno de sus niveles viene guardado en un paquete con otras matrices para simplificar su escritura, lectura y corrección.

Siendo este primer nivel el correspondiente a las bahías duplas:

```
1 Contenedor[][][] matrizBeatrizB = {matrizBeatrizB_Db2,
    matrizBeatrizB_Db6, matrizBeatrizB_Db10, matrizBeatrizB_Db14,
    matrizBeatrizB_Db18, matrizBeatrizB_Db22, matrizBeatrizB_Db26,
    matrizBeatrizB_Db30, matrizBeatrizB_Db34};
```

Para el caso de la bahía dupla 10:

```
1 Contenedor[][][] matrizBeatrizB_Db10 = {matrizBeatrizB_bahia_10_9,
    matrizBeatrizB_bahia_10_11};
```

Dentro de la bahía simple 9:

```
1 Contenedor[][][] matrizBeatrizB_bahia_10_9 = {matrizBeatrizB_bodega_10_9,
    matrizBeatrizB_cubierta_10_9};
```

Dentro de su cubierta ya están implementados directamente las filas y columnas, siendo HUECO, como visto antes, un contenedor que representa la disponibilidad de espacio:

```
1 Contenedor[][][] matrizBeatrizB_cubierta_10_9 = {{HUECO, HUECO, HUECO, HUECO},
    {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
    HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO},
    {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
    HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO},
    {HUECO, HUECO, HUECO, HUECO}};
```

Por lo que los barcos codificados vienen vacíos por defecto.

El conjunto de todos los espacios de Beatriz B que conforman la matriz es el siguiente:

```
1 Contenedor[][][] matrizBeatrizB_bodega_2_1 = {{ESTRUCTURAL, ESTRUCTURAL,
    ESTRUCTURAL, ESTRUCTURAL, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, ESTRUCTURAL,
    HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {ESTRUCTURAL,
    ESTRUCTURAL, ESTRUCTURAL, HUECO, HUECO}, {ESTRUCTURAL, ESTRUCTURAL,
    ESTRUCTURAL, ESTRUCTURAL, HUECO}};
    Contenedor[][][] matrizBeatrizB_cubierta_2_1 = {{HUECO, HUECO, HUECO},
2 {HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO,
    HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO, HUECO,
    HUECO}};
3 Contenedor[][][] matrizBeatrizB_bahia_2_1 = {matrizBeatrizB_bodega_2_1,
    matrizBeatrizB_cubierta_2_1};
4 Contenedor[][][] matrizBeatrizB_Db2 = {matrizBeatrizB_bahia_2_1,
    null};
5
6 Contenedor[][][] matrizBeatrizB_bodega_6_5 = {{ESTRUCTURAL, ESTRUCTURAL,
    ESTRUCTURAL, ESTRUCTURAL, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, ESTRUCTURAL,
    HUECO, HUECO}, {ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
    HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {ESTRUCTURAL,
```

```

HUECO, HUECO, HUECO, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, ESTRUCTURAL, HUECO,
HUECO}, {ESTRUCTURAL, ESTRUCTURAL, ESTRUCTURAL, ESTRUCTURAL, HUECO}};
7   Contenedor[][] matrizBeatrizB_cubierta_6_5 = {{HUECO, HUECO, HUECO},
{HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO}, {HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO}};
8   Contenedor[][][] matrizBeatrizB_bahia_6_5 = {matrizBeatrizB_bodega_6_5,
matrizBeatrizB_cubierta_6_5};
9   Contenedor[][] matrizBeatrizB_bodega_6_7 = {{ESTRUCTURAL, ESTRUCTURAL,
ESTRUCTURAL, ESTRUCTURAL, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, HUECO, HUECO,
HUECO}, {ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {ESTRUCTURAL, HUECO,
HUECO, HUECO, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, HUECO, HUECO, HUECO},
{ESTRUCTURAL, ESTRUCTURAL, ESTRUCTURAL, ESTRUCTURAL, HUECO}};
10  Contenedor[][] matrizBeatrizB_cubierta_6_7 = {{HUECO, HUECO, HUECO,
HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}};
11  Contenedor[][][] matrizBeatrizB_bahia_6_7 = {matrizBeatrizB_bodega_6_7,
matrizBeatrizB_cubierta_6_7};
12  Contenedor[][][] matrizBeatrizB_Db6 = {matrizBeatrizB_bahia_6_5,
matrizBeatrizB_bahia_6_7};
13
14  Contenedor[][] matrizBeatrizB_bodega_10_9 = {{ESTRUCTURAL, ESTRUCTURAL,
HUECO, HUECO, HUECO}, {ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO},
{ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, HUECO,
HUECO, HUECO}};
15  Contenedor[][] matrizBeatrizB_cubierta_10_9 = {{HUECO, HUECO, HUECO,
HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}};
16  Contenedor[][][] matrizBeatrizB_bahia_10_9 =
{matrizBeatrizB_bodega_10_9, matrizBeatrizB_cubierta_10_9};
17  Contenedor[][] matrizBeatrizB_bodega_10_11 = {{ESTRUCTURAL, ESTRUCTURAL,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, HUECO, HUECO, HUECO}};
18  Contenedor[][] matrizBeatrizB_cubierta_10_11 = {{HUECO, HUECO, HUECO,
HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO}};
19  Contenedor[][][] matrizBeatrizB_bahia_10_11 =
{matrizBeatrizB_bodega_10_11, matrizBeatrizB_cubierta_10_11};
20  Contenedor[][][] matrizBeatrizB_Db10 = {matrizBeatrizB_bahia_10_9,
matrizBeatrizB_bahia_10_11};
21
22  Contenedor[][] matrizBeatrizB_bodega_14_13 = {{ESTRUCTURAL, HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO}, {ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO}};

```



```

HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}};
56     Contenedor[][][] matrizBeatrizB_bahia_30_29 =
{matrizBeatrizB_bodega_30_29, matrizBeatrizB_cubierta_30_29};
57     Contenedor[][] matrizBeatrizB_bodega_30_31 = {{ESTRUCTURAL, ESTRUCTURAL,
HUECO, HUECO, HUECO}, {ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO},
{ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO}, {ESTRUCTURAL, ESTRUCTURAL, HUECO,
HUECO, HUECO}};
58     Contenedor[][] matrizBeatrizB_cubierta_30_31 = {{HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO}};
59     Contenedor[][][] matrizBeatrizB_bahia_30_31 =
{matrizBeatrizB_bodega_30_31, matrizBeatrizB_cubierta_30_31};
60     Contenedor[][][] matrizBeatrizB_Db30 = {matrizBeatrizB_bahia_30_29,
matrizBeatrizB_bahia_30_31};
61
62     Contenedor[][] matrizBeatrizB_cubierta_34_33 = {{ESTRUCTURAL, HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO, HUECO},
{HUECO, HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO, HUECO},
{HUECO, HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO,
HUECO, HUECO}, {ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO, HUECO}};
63     Contenedor[][][] matrizBeatrizB_bahia_34_33 = {null,
matrizBeatrizB_cubierta_34_33};
64     Contenedor[][] matrizBeatrizB_cubierta_34_35 = {{ESTRUCTURAL, HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO, HUECO},
{HUECO, HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO,
HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO,
HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO, HUECO, HUECO},
{HUECO, HUECO, HUECO, HUECO, HUECO, HUECO}, {HUECO, HUECO, HUECO, HUECO,
HUECO, HUECO}, {ESTRUCTURAL, HUECO, HUECO, HUECO, HUECO, HUECO}};
65     Contenedor[][][] matrizBeatrizB_bahia_34_35 = {null,
matrizBeatrizB_cubierta_34_35};
66     Contenedor[][][] matrizBeatrizB_Db34 = {matrizBeatrizB_bahia_34_33,
matrizBeatrizB_bahia_34_35};

```

4.4. PAQUETE SOPORTE

Este paquete contiene solo una clase, **Listados**, donde se alojan los métodos encargados de generar contenedores y los objetos que contienen los puertos y empresas.

Algunos de los objetos correspondientes a los puertos:

```
1 final public Puerto THLCH = new Puerto("Laem Chabang", "THLCH",  
    "Tailandia");  
2 final public Puerto IDJKT = new Puerto("Tanjung Priok", "IDJKT",  
    "Indonesia");  
3 final public Puerto USNYC = new Puerto("Nueva York-Nueva Jersey", "USNYC",  
    "Estados Unidos");  
4 final public Puerto LKCMB = new Puerto("Colombo", "LKCMB", "Sri Lanka");  
5 final public Puerto VNSGN = new Puerto("Ciudad de Ho Chi Minh", "VNSGN",  
    "Vietnam");  
6 final public Puerto CNYIK = new Puerto("Yingkou", "CNYIK", "China");  
7 final public Puerto DEBRE = new Puerto("Bremen", "DEBRE", "Alemania");  
8 final public Puerto INNSA = new Puerto("Jawaharlal Nehru", "INNSA",  
    "India");  
9 final public Puerto ESVLC = new Puerto("Valencia", "ESVLC", "España");  
10 final public Puerto PHMNL = new Puerto("Manila", "PHMNL", "Filipinas");  
11 final public Puerto CNTAC = new Puerto("Taicang", "CNTAC", "China");  
12 final public Puerto GRPIR = new Puerto("El Pireo", "GRPIR", "Grecia");  
13 final public Puerto ESALG = new Puerto("Algeciras", "ESALG", "España");
```

Algunos de los objetos correspondientes a las empresas, adicionalmente al objeto empresa le acompañan dos objetos Color, que se utilizan en el visor:

```
1 final private Color fueMSC = new Color(46, 59, 78);  
2 final private Color fonMSC = new Color(195, 164, 81);  
3 final public Empresa MSC = new Empresa("Mediterranean Shipping Company  
    S.A.", "MSC", "Suiza", fueMSC, fonMSC, 47.17, 'b');  
4 final private Color fueCSN = new Color(230, 237, 243);  
5 final private Color fonCSN = new Color(1, 140, 223);  
6 final public Empresa CSN = new Empresa("COSCO", "CSN", "China", fueCSN,  
    fonCSN, 41.06, 'b');  
7 final private Color fueCMA = new Color(215, 214, 210);  
8 final private Color fonCMA = new Color(5, 5, 255);  
9 final public Empresa CMA = new Empresa("CMA CGM Group", "CMA", "Francia",  
    fueCMA, fonCMA, 35.14, 'f');  
10 final private Color fueHBL = new Color(19, 37, 103);  
11 final private Color fonHBL = new Color(239, 95, 8);  
12 final public Empresa HBL = new Empresa("Hapag-Lloyd", "HBL", "Alemania",  
    fueHBL, fonHBL, 31.61, 'f');  
13 final private Color fueONE = new Color(204, 210, 207);  
14 final private Color fonONE = new Color(193, 0, 109);  
15 final public Empresa ONE = new Empresa("Ocean Network Express", "ONE",  
    "Singapur", fueONE, fonONE, 28.57, 'b');
```

Estos objetos se agrupan en listados que de igual manera que el de mercancías peligrosas, se utilizarán en la generación de contenedores.

Listado de puertos:

```
1 final public Puerto[] listaPuertos = {CNSHG, SGSIN, CNNGB, CNSNZ, CNCAN,  
KRPUS, HKHKG, CNQDG, CNTXG, AEDXB, NLRTM, MYPKG, BEANR, CNXMG, TWKHH, CNDLC,  
USLAX, MYTPP, DEHAM, USLGB, THLCH, IDJKT, USNYC, LKCMB, VNNGN, CNYIK, DEBRE,  
INNSA, ESVLC, PHMNL, CNTAC, GRPIR, ESALG, CNLYG, JPTYO, INMUN, USSAV, PAONX,  
BRSSZ, SAJED, CNRZH, GBFXT, IDSUB, USSEA, VNCMT, CNDGG, MAPTM, ESBCN, CNFOC,  
CAVAN, OMSLL, MTMAR, CNNKG, TRAMR, KRINC, MXZLO, EGPSD, JPYOK, AUMEL, CNYTG,  
CNTGS, ZADUR, JPUKB, BDCGP, COCTG, JPNGO, FRLEH, USORF, USHOU, GBLON, ITGOA,  
AUSYD, USOAK, PABLB, JPOSA, KRKAN, CNQZJ, PECALL, ITGIT, USCHS, CNZUH, SAKAC,  
PKKHI, RULED, ECGYE, IRBND, AEKLF, GBSOU, PLGDN, CNHAK, JMKIN, ARBUE, PTSIE,  
TWTXG, AEAUH, CNZAP, TRMER, CAMTR, CLSAI, TWTPPE, DZORAN, ARROS, ARBQS, AUBNE,  
CVRAI, CMDLA, ESBIO, ESVIL, ESCAD, ESLPA, ESSCT, ESPMI, ESALC, ESFUE, ESACE,  
ESCAS, ESSDR, ESAGP, ESGO, ESGIJ, ESTAR, ESFRO, ESCEU, ESMLN, ESSVQ, ESCAR,  
ESHUV, FRIRK, FRMRS, RUVVO, GQSSG, GQBSG, GWOXB, USMIA, MACAS, MRNKC, MRNDB,  
NGLOS, NCNOU, NZWLK, NZAKL, PGPOM, PTSET, PTLEI, SNDKR, SOMGQ, TGLFW, TRIZM,  
TRIST, UYMVD};
```

Listado de empresas:

```
1 final public Empresa[] listaEmpresas = {TEX, TTN, MAE, MSC, CSN, CMA, HBL,  
ONE, EMC, ALM, TCS, CIL, EUR, YML, PRK, THI, RWT, VTG, RAW, PIL, HMM, ARK,  
ZIM, WHL, ZGR, IRS, GCN, KMT, SIT, TSL, SUD, SMC, SNT, MSK, SKL, REG, SPN,  
TLC, CIC, JSS, GXL, MAT, NID, UAC, BOL, BBU, HCS, OPD, TMY, MOL};
```

4.4.1 Generación de contenedores

En el paquete soporte se encuentra el método encargado de generar packing lists. La razón por la que se creó era para poder disponer de una gran variedad de rutas y packing lists sin tener que escribirlos de forma manual.

El método `azarPOL` se utiliza para obtener puertos de partida de contenedores de múltiples orígenes, se utiliza para generar el packing list de la carga previa del barco, que es desconocida y simula la situación inicial en la que se encuentra el barco. Tiene como parámetros de entrada el listado de puertos que el barco va a recorrer.

```
1 public Puerto azarPOL(ArrayList<Puerto> puertosDestino) {
2     Puerto POL = new Puerto();
3     Boolean e = true;
4     while (e) {
5         POL = listaPuertos[(int) (Math.random() * listaPuertos.length)];
6         for (int i = 0; i < puertosDestino.size(); i++) {
7             if (POL == puertosDestino.get(i)) {
8                 e = true;
9                 break;
10            } else {
11                e = false;
12            }
13        }
14    }
15    return POL;
16 }
```

El siguiente es el método encargado de generar contenedores:

`generadorListaContenedores`, que cuenta con los siguientes parámetros de entrada:

- listado (`int`): Posición en el que se encuentra el packing list.
- espacio (`int`): Número mínimo de contenedores que va a haber por packing list.
- coefExtra (`double`): Porcentaje de contenedores de más que van a ponerse en los packing list. Sin implantación en la interfaz.
- empresa (`Empresa`): Empresa prioritaria, es la que tiene un mayor número de contenedores que el resto, generalmente la naviera. Sin

implantación en la interfaz.

- `porcEmpresa` (`double`): Porcentaje de contenedores que pertenecen a la empresa prioritaria. Sin implantación en la interfaz.
- `POL` (`Puerto`): Puerto de carga de los contenedores del listado.
- `puertosDestino` (`ArrayList<Puerto>`): Listado de puertos de destino.

Sigue los siguientes pasos:

1. Calcula el número de contenedores que tiene que generar.
2. Calcula el número de contenedores que van a ser de la empresa prioritaria.
3. Calcula el número de contenedores que van a ser de terceros.
4. Crea una lista de contenedores vacía.
5. Genera los contenedores:
 - a. Si es una carga previa (listado 0): Realizará el mismo procedimiento que una carga posterior (b.) con la diferencia de que el POL será al azar dado por `azarPOL`.
 - b. Van generando contenedores, dando prioridad a la empresa seleccionada en los parámetros. Si es una de las empresas que van al azar, la escogerán entre las empresas alojadas en su `listado`. Debido al atributo de empresa `porcen`, habrá empresas con mas probabilidad de salir que otras.
 - c. El POL será el que se corresponda a su número de `listado`.
 - d. En base a la posición en la que se encuentre `listado`, los POD variarán, un POD no puede ser de un puerto que no se vaya a recorrer.
 - e. Debido a que el contenedor accede a su parámetro `conhab`, el cual indicará que tipo de contenedores la empresa puede tener.
 - f. En base al tipo de contenedor podemos saber las capacidades del contenedor en relación con pesos y mercancías peligrosas. Estos datos se han establecido consultando múltiples fuentes, cabe mencionar que muchas de éstas se contradicen puesto que cada fabricante, pese a

seguir los estándares de la norma tienen resultados diferentes en sus productos, los datos que se emiten son una estimación de lo recopilado.

- g. La matrícula del contenedor "??U", se actualizará:
Estableciéndose el código bic de la empresa, un número de serie y el dígito de control.
- h. Dependiendo del tipo de contenedor habrá:
 - Una probabilidad del 2% que un contenedor general tenga mercancía peligrosa.
 - Una probabilidad del 4% si es un contenedor general ventilado tenga mercancía peligrosa.
 - Una probabilidad del 8% si es un contenedor cisterna tenga mercancía peligrosa.

```

1 public Contenedor[] generadorListaContenedores(int listado, int espacio,
2 double coefExtra, Empresa empresa, double porcEmpresa, Puerto POL,
3 ArrayList<Puerto> puertosDestino) {
4     int reserva = (int) (coefExtra * espacio); //Indica el número de
5     contenedores de más que se van a hacer.
6     reserva = reserva + espacio;
7     int conPropio = (int) (porcEmpresa * reserva); //n° de contenedores
8     que por defecto serán de la empresa del barco
9     int conExtra = (int) ((1 - porcEmpresa) * reserva); //n° de
10    contenedores que pueden ser de cualquier tipo de empresa
11    reserva = conPropio + conExtra;
12    Contenedor[] listaContenedores = new Contenedor[reserva];
13    for (int i = 0; i < reserva; i++) {
14        int pos = (int) (Math.random() *
15        (puertosDestino.size())); //Destino al azar.
16        while (puertosDestino.get(pos) == POL) { //Mientras el destino al
17        azar sea el mismo que el puerto en el que nos encontremos...
18            pos = (int) (Math.random() *
19            puertosDestino.size()); //...habrá que cambiar de destino al azar.
20        }
21        if (!POL.getNombre().equals("Inicio")) {
22            if (i < conPropio) {
23                listaContenedores[i] = new Contenedor(listado, i + 1,
24                "??U" + String.format("%06d", (int) (Math.random() * 999999)) + "?", "",
25                POL, puertosDestino.get(pos), 0, 0, 0, 0, lisM.MMPP_0, empresa);
26            } else {
27                listaContenedores[i] = new Contenedor(listado, i + 1,
28                "??U" + String.format("%06d", (int) (Math.random() * 999999)) + "?", "",
29                POL, puertosDestino.get(pos), 0, 0, 0, 0, lisM.MMPP_0, aleatorioEmpresa());
30            }
31        } else {
32            if (i < conPropio) {
33                listaContenedores[i] = new Contenedor(listado, i + 1,
34                "??U" + String.format("%06d", (int) (Math.random() * 999999)) + "?", "",
35                azarPOL(puertosDestino), puertosDestino.get(pos), 0, 0, 0, 0, lisM.MMPP_0,

```



```

empresa);
22         } else {
23             listaContenedores[i] = new Contenedor(listado, i + 1,
"???U" + String.format("%06d", (int) (Math.random() * 999999)) + "?", "",
azarPOL(puertosDestino), puertosDestino.get(pos), 0, 0, 0, 0, lisM.MMPP_0,
aleatorioEmpresa());
24         }
25     }
26     listaContenedores[i].aleatorioDYTconhab();
27     listaContenedores[i].actualizarContenedor();
28     if(Math.random() < 0.1) {
29         listaContenedores[i].setPesoCarga(0);
30     }
31     listaContenedores[i].calcularVGM();
32     if (listaContenedores[i].getPesoCarga() != 0) {
33         if (listaContenedores[i].getDyt().charAt(2) == 'G') { //Si es
un contenedor de uso general
34             double prob = Math.random();
35             if (prob < 0.05) {
36                 Mmpp peligroso = lisM.LISTA_IMDG[(int)
(Math.random() * lisM.LISTA_IMDG.length)];
37                 while (peligroso.getClase_3().equals("2.1") ||
peligroso.getClase_3().equals("2.2") || peligroso.getClase_3().equals("2.3")
|| peligroso.getClase_3().equals("3")) {
38                     peligroso = lisM.LISTA_IMDG[(int) (Math.random()
* lisM.LISTA_IMDG.length)];
39                 }
40                 listaContenedores[i].setMmpp(peligroso);
41             }
42         } else if (listaContenedores[i].getDyt().charAt(2) == 'V')
//Si es un contenedor de uso general ventilado
43             double prob = Math.random();
44             if (prob < 0.1) {
45                 Mmpp peligroso = lisM.LISTA_IMDG[(int)
(Math.random() * lisM.LISTA_IMDG.length)];
46                 while (peligroso.getClase_3().equals("2.1") ||
peligroso.getClase_3().equals("2.2") || peligroso.getClase_3().equals("2.3")
|| peligroso.getClase_3().equals("3")) {
47                     peligroso = lisM.LISTA_IMDG[(int) (Math.random()
* lisM.LISTA_IMDG.length)];
48                 }
49                 listaContenedores[i].setMmpp(peligroso);
50             }
51         } else if (listaContenedores[i].getDyt().charAt(2) == 'T')
//Si es un contenedor cisterna
52             double prob = Math.random();
53             if (prob < 0.4) {
54                 Mmpp peligroso = lisM.LISTA_IMDG[(int)
(Math.random() * lisM.LISTA_IMDG.length)];
55                 while (!(peligroso.getClase_3().equals("2.1") ||
peligroso.getClase_3().equals("2.2") || peligroso.getClase_3().equals("2.3")
|| peligroso.getClase_3().equals("3"))) {
56                     peligroso = lisM.LISTA_IMDG[(int) (Math.random()
* lisM.LISTA_IMDG.length)];
57                 }
58                 listaContenedores[i].setMmpp(peligroso);
59             }

```

```

60         }
61     }
62 }
63     ArrayList<Contenedor> desorden = new
        ArrayList<Contenedor>(Arrays.asList(listaContenedores));
64     Collections.shuffle(desorden);
65     desorden.toArray(listaContenedores);
66     return listaContenedores;
67 }

```

Vamos a poner en funcionamiento este método, crearemos una lista de 18 contenedores para una ruta entre tres puertos, Mallorca, Algeciras y Tenerife. Donde la naviera prioritaria será Grimaldi, suyos serán la mitad de los contenedores y el puerto en el que nos localizamos es Mallorca.

Los resultados escritos en forma de objeto impresos por consola son los siguientes:

```

Contenedor _1_n1 = new Contenedor(1, 1, "TEXU2286377", "42G1", lis.ESLPA,
lis.ESSCT, 3750.0, 28720.0, 10079.64, 13830.0, lisM.MMPP_0, lis.TEX);
Contenedor _1_n2 = new Contenedor(1, 2, "MSCU6594014", "25G2", lis.ESLPA,
lis.ESSCT, 2340.0, 28180.0, 19617.61, 21958.0, lisM.MMPP_0, lis.MSC);
Contenedor _1_n3 = new Contenedor(1, 3, "GCNU9203465", "25G0", lis.ESLPA,
lis.ESBCN, 2340.0, 28180.0, 8504.92, 10845.0, lisM.MMPP_0, lis.GCN);
Contenedor _1_n4 = new Contenedor(1, 4, "TEXU6826196", "45V3", lis.ESLPA,
lis.ESBCN, 3900.0, 28600.0, 26746.59, 30647.0, lisM.MMPP_0, lis.TEX);
Contenedor _1_n5 = new Contenedor(1, 5, "GCNU8732464", "22G0", lis.ESLPA,
lis.ESSCT, 2300.0, 25000.0, 9374.19, 11674.0, lisM.MMPP_1788, lis.GCN);
Contenedor _1_n6 = new Contenedor(1, 6, "TEXU6081842", "45B3", lis.ESLPA,
lis.ESBCN, 3900.0, 28600.0, 13319.2, 17219.0, lisM.MMPP_0, lis.TEX);
Contenedor _1_n7 = new Contenedor(1, 7, "GCNU0062999", "42G2", lis.ESLPA,
lis.ESSCT, 3750.0, 28720.0, 15513.79, 19264.0, lisM.MMPP_0, lis.GCN);
Contenedor _1_n8 = new Contenedor(1, 8, "GCNU6389525", "22G2", lis.ESLPA,
lis.ESSCT, 2300.0, 25000.0, 15824.89, 18125.0, lisM.MMPP_0, lis.GCN);
Contenedor _1_n9 = new Contenedor(1, 9, "MAEU2824193", "45G0", lis.ESLPA,
lis.ESBCN, 3900.0, 28600.0, 16795.33, 20695.0, lisM.MMPP_0, lis.MAE);
Contenedor _1_n10 = new Contenedor(1, 10, "TTNU9970462", "42G0", lis.ESLPA,
lis.ESSCT, 3750.0, 28720.0, 17229.63, 20980.0, lisM.MMPP_2605, lis.TTN);
Contenedor _1_n11 = new Contenedor(1, 11, "EURU9653053", "22T1", lis.ESLPA,
lis.ESSCT, 3070.0, 27410.0, 19737.25, 22807.0, lisM.MMPP_0, lis.EUR);
Contenedor _1_n12 = new Contenedor(1, 12, "GCNU5510479", "22G1", lis.ESLPA,
lis.ESBCN, 2300.0, 25000.0, 6092.54, 8393.0, lisM.MMPP_0, lis.GCN);
Contenedor _1_n13 = new Contenedor(1, 13, "GCNU9735257", "22G1", lis.ESLPA,
lis.ESSCT, 2300.0, 25000.0, 1626.48, 3926.0, lisM.MMPP_0, lis.GCN);
Contenedor _1_n14 = new Contenedor(1, 14, "GCNU6680000", "22G1", lis.ESLPA,
lis.ESBCN, 2300.0, 25000.0, 11264.76, 13565.0, lisM.MMPP_0, lis.GCN);
Contenedor _1_n15 = new Contenedor(1, 15, "MAEU4041622", "42G0", lis.ESLPA,
lis.ESBCN, 3750.0, 28720.0, 23217.62, 26968.0, lisM.MMPP_0, lis.MAE);
Contenedor _1_n16 = new Contenedor(1, 16, "GCNU7800667", "45G1", lis.ESLPA,
lis.ESSCT, 3900.0, 28600.0, 25062.4, 28962.0, lisM.MMPP_0, lis.GCN);
Contenedor _1_n17 = new Contenedor(1, 17, "TEXU8393217", "45V1", lis.ESLPA,
lis.ESSCT, 3900.0, 28600.0, 14928.3, 18828.0, lisM.MMPP_0, lis.TEX);
Contenedor _1_n18 = new Contenedor(1, 18, "GCNU7668550", "42G2", lis.ESLPA,
lis.ESSCT, 3750.0, 28720.0, 16629.06, 20379.0, lisM.MMPP_0, lis.GCN);

```

Estos contenedores pueden pasarse por un toString para comprender la información almacenada, veamos como ejemplo el tercer, cuarto y quinto contenedor.

Contenedor GCNU9203465

Listado 1, reserva nº00003
Matrícula: GCNU9203465
Dimensión y Tipo: 25G0

Es un contenedor para uso general sin ventilación con abertura(s) en una o las dos extremidades;

>Mide:

2.895 metros de alto,
6.068 metros de largo,
2.438 metros de ancho.

Se embarcó en Las Palmas de Gran Canaria(España).

Se desembarca en Barcelona(España).

Tara: 2340.0, Peso útil: 28180.0, Peso carga: 8504.92, que en total da un VGM de 10845.0.

>No contiene mercancías peligrosas.

El contenedor es propiedad de Grimaldi.

Contenedor TEXU6826196

Listado 1, reserva nº00004
Matrícula: TEXU6826196
Dimensión y Tipo: 45V3

Es un contenedor para uso general ventilado con código de características relativas sin asignar;

>Mide:

2.895 metros de alto,
12.192 metros de largo,
2.438 metros de ancho.

Se embarcó en Las Palmas de Gran Canaria(España).

Se desembarca en Barcelona(España).

Tara: 3900.0, Peso útil: 28600.0, Peso carga: 26746.59, que en total da un VGM de 30647.0.

>No contiene mercancías peligrosas.

El contenedor es propiedad de Textainer Group Holdings.

Contenedor GCNU8732464

Listado 1, reserva nº00005
Matrícula: GCNU8732464
Dimensión y Tipo: 22G0

Es un contenedor para uso general sin ventilación con abertura(s) en una o las dos extremidades;

>Mide:

2.591 metros de alto,
6.068 metros de largo,
2.438 metros de ancho.

Se embarcó en Las Palmas de Gran Canaria(España).

Se desembarca en Santa Cruz de Tenerife(España).

Tara: 2300.0, Peso útil: 25000.0, Peso carga: 9374.19, que en total da un VGM de 11674.0.

>Contiene mercancías peligrosas:

nº un: 1788

ÁCIDO BROMHÍDRICO

Clase 8: Sustancias corrosivas.

Riesgo secundario: -

No es un contaminante del mar.

Grupo de embalaje/envase: II, Sustancias que presentan peligrosidad media.

Criterios de estiba y manejo: En cubierta solamente.

Criterios de segregación: -

El contenedor es propiedad de Grimaldi.

4.5. PAQUETE INTERFAZ

Dentro del paquete interfaz se encuentra el código dedicado a crear los menús y dar soporte gráfico al programa. Aquí se encuentra el método NuevaRuta, que es el encargado de la ejecución de todo el código.

No se entrará a describir en profundidad la elaboración de menús. Son una serie de clases encargadas de crear interfaces gráficas, con botones, casillas y tablas. Su finalidad es aportar al usuario un entorno amigable en el que trabajar la información. Gracias al soporte dado por Netbeans, mucho de su código ha sido generado y el tiempo para desarrollarlas no fue excesivo.

Dentro de este paquete hay dos clases que sirven para dar soporte a las interfaces **Dibujo** y **Estadísticas**.

4.5.1. CLASE ESTADÍSTICAS

Cuando se introduce una ruta y un barco, los algoritmos de carga procederán a realizar los diferentes planos de carga y por ello a producir mucha información. Esta clase se encarga de procesar y almacenar toda la información producida.

Esta clase importa la librería de Oracle `Serializable`, necesaria para el guardado de la información.

Declaración de atributos.

```
1 String nombre;  
2 Viaje via;  
3 int[] concar;  
4 int[] conpel;  
5 int[] convac;  
6 int[] pestot;  
7 int[] nuent;  
8 int[] nusal;  
9 int[][][] conton;  
10 Object[][][] arrcon;
```

Donde los datos representan:

- nombre (`String`): Nombre del conjunto de datos.
- via (`Viaje`): Viaje realizado (Recuerde que consta de tres atributos: Barco, ruta y algoritmo de carga).
- concar (`int[]`): Número de contenedores cargados en cada posición del viaje.
- conpel (`int[]`): Número de contenedores peligrosos cargados en cada posición del viaje.
- convac (`int[]`): Número de contenedores vacíos en cada posición del viaje.
- pestot (`int[]`): Peso total de la carga en cada posición del viaje.
- nuent (`int[]`): Número de contenedores que han entrado en el barco dada una posición.
- nusal (`int[]`): Número de contenedores que han salido en el barco dada una posición.

- `conton (int[] [] [])`: Pesos por bahías en cada posición del viaje.
- `arrcon (Object[] [] [])`: Lista de contenedores almacenados por cada posición del viaje.

Constructor.

```

1 public Estadisticas(String nombre, Viaje via) {
2     this.nombre = nombre;
3     this.via = via;
4     Druta = this.getVia().getCargas().length;
5     listaContenedores = new ArrayList<ArrayList<Contenedor>>(Druta);
6     for (int i = 0; i < Druta; i++) {
7         listaContenedores.add(via.getCargas()[i].contenedoresEnBarco());
8     }
9     concar = this.datoConcar();
10    conpel = this.datoConpel();
11    convac = this.datoConvac();
12    pestot = this.datoPestot();
13    nuent = this.datoNuent();
14    nusal = this.datoNusal();
15    conton = this.datoConton();
16    arrcon = this.datoArrcon();
17 }

```

Como puede observarse, la clase estadísticas recopila todos sus datos a partir de Viaje. En los siguientes métodos de ejemplo puede observarse como lo hace.

Métodos secundarios de interés.

Para obtener el nº de contenedores cargados por posición de la ruta para el atributo concar:

```

1 protected int[] datoConcar() {
2     int[] concar = new int[Druta];
3     for (int i = 0; i < Druta; i++) {
4         concar[i] = listaContenedores.get(i).size();
5     }
6     return concar;
7 }

```

Para obtener el nº de contenedores que se descargan en cada puerto para el atributo nusal:

```

1 protected int[] datoNusal() {
2     int[] nusal = new int[Druta];
3     for (int i = 0; i < Druta; i++) {
4         if (i != 0) {

```

```

5         nusal[i] =
    this.getVia().getBarco().contenedoresFuera(this.getVia().getRuta().getPuertos(
    ).get(i).getLocode(), listaContenedores.get(i - 1));
6     }
7 }
8     return nusal;
9 }

```

En el menú habrá una tabla que muestre todos los contenedores que se encuentran cargados en el barco (según en que posición de la ruta se encuentre), para ello recopilarán la siguiente información:

```

1 final protected String[] datosColumnas = {"Posición", "ID", "POL", "POD",
    "Peso tara", "Peso útil", "Peso carga", "VGM", "MMPP (n°UN)", "Alt.", "Lar.",
    "Anc.", "Tipo", "Empresa", "Lista"};

```

En este método se recopilan los datos de arrcon, se serán mostrados en una tabla en el menú principal:

```

1 protected Object[][][] datoArrcon() {
2     Object[][][] matrizO = new
    String[this.via.getCargas().length][[]]; //Se crea matrizO que es un array
    de objetos de tres dimensiones, la primera representa el barco en cada una
    de sus cargas, la segunda el listado de contenedores que contiene, la
    tercera lleva la información del contenedor.
3     int dimMax = this.via.getBarco().alturaMasLargaDelBarco();
4     for (int pos = 0; pos < this.via.getCargas().length; pos++) { //Se
    va de carga en carga
5         ArrayList<Contenedor> conArray =
    this.via.getCargas()[pos].contenedoresEnBarco(); //Se extraen todos los
    contenedores que contiene la carga y se guardan en un arrayList
6         matrizO[pos] = new String[conArray.size()][[]]; //Se inicializa la
    segunda dimensión de matrizO
7         for (int posArr = 0; posArr < conArray.size(); posArr++)
    { //Ahora se recorre el listado de posiciones de los contenedores
8             matrizO[pos][posArr] = new
    String[datosColumnas.length]; //El contenido del array será el mismo número
    de columnas de la tabla.
9             for (int ii = 0; ii < datosColumnas.length; ii++) { //Por
    último se recorre la lista objeto a objeto.
10                 if (ii == 0 || ii == 1) {
11                     String res1 = "";
12                     String res = "";
13                     for (int i = 0; i <
    this.via.getCargas()[pos].getMatriz().length; i++) {
14                         for (int j = 0; j <
    this.via.getCargas()[pos].getMatriz()[i].length; j++) {
15                             if
    (this.via.getCargas()[pos].getMatriz()[i][j] != null) {
16                                 for (int k = 0; k <
    this.via.getCargas()[pos].getMatriz()[i][j].length; k++) {
17                                     if
    (this.via.getCargas()[pos].getMatriz()[i][j][k] != null) {

```

```

18         for (int m = 0; m <
this.via.getCargas()[pos].getMatriz()[i][j][k].length; m++) {
19             for (int n = 0; n <
this.via.getCargas()[pos].getMatriz()[i][j][k][m].length; n++) {
20                 if
(this.via.getCargas()[pos].getMatriz()[i][j][k][m][n].getId().equals(conArr
ay.get(posArr).getId())) {
21                     if (ii == 0) {
22                         res1 = "[" + i
+ "[" + j + "[" + k + "[" + m + "[" + n + "];
23                     } else {
24                         boolean
esBodega = false;
25                         if (k == 0) {
26                             esBodega =
true;
27                         }
28                         if
(conArray.get(posArr).longitudContenedor() >= 12.192) {//ES UN CONTENEDOR
QUE OCUPA UNA DUPLA
29                             res = res +
String.format("%02d", this.via.getBarco().conversion_posarrayAposbahia(i));
30                             } else {//ES UN
CONTENEDOR QUE OCUPA UNA BAHÍA
31                                 int e = 1;
32                                 if (j == 0)
33                                 {
34                                     e = -1;
35                                 }
36                                 res = res +
String.format("%02d", (e +
this.via.getBarco().conversion_posarrayAposbahia(i)));
37                             }
38                             res = res +
String.format("%02d", this.via.getBarco().conversion_posarrayAposfila(m,
this.via.getCargas()[pos].getMatriz()[i][j][k].length));
39                             if (esBodega) {
40                                 res = res +
String.format("%02d", 2 * (n - 1);
41                             } else {
42                                 res = res +
String.format("%02d", this.via.getBarco().conversion_posarrayAposnivel(n,
esBodega, this.via.getCargas()[pos].getMatriz()[i][j][k][m].length));
43                             }
44                             break;
45                         }
46                     }
47                 }
48             }
49         }
50     }
51 }
52 }
53 }
54
55     if (ii == 0) {
56         matrizO[pos][posArr][ii] = res1;

```



```

57         } else {
58             matrizO[pos][posArr][ii] = res;
59         }
60     } else if (ii == 2) {
61         matrizO[pos][posArr][ii] =
conArray.get(posArr).getId();
62     } else if (ii == 3) {
63         matrizO[pos][posArr][ii] =
conArray.get(posArr).getPol().getLocode();
64     } else if (ii == 4) {
65         matrizO[pos][posArr][ii] =
conArray.get(posArr).getPod().getLocode();
66     } else if (ii == 5) {
67         matrizO[pos][posArr][ii] =
conArray.get(posArr).getPesoTara() + " kgs.";
68     } else if (ii == 6) {
69         matrizO[pos][posArr][ii] =
conArray.get(posArr).getPesoUtil() + " kgs.";
70     } else if (ii == 7) {
71         matrizO[pos][posArr][ii] =
conArray.get(posArr).getPesoCarga() + " kgs.";
72     } else if (ii == 8) {
73         matrizO[pos][posArr][ii] =
conArray.get(posArr).getVgm() + " kgs.";
74     } else if (ii == 9) {
75         matrizO[pos][posArr][ii] =
conArray.get(posArr).getMmpp().getNoex_2();
76     } else if (ii == 10) {
77         matrizO[pos][posArr][ii] =
conArray.get(posArr).alturaContenedorTX() + " m";
78     } else if (ii == 11) {
79         matrizO[pos][posArr][ii] =
conArray.get(posArr).longitudContenedorTX() + " m";
80     } else if (ii == 12) {
81         matrizO[pos][posArr][ii] =
conArray.get(posArr).anchuraContenedorTX() + " m";
82     } else if (ii == 13) {
83         matrizO[pos][posArr][ii] = "C" +
conArray.get(posArr).tipoContenedor().substring(1).substring(1);
84     } else if (ii == 14) {
85         matrizO[pos][posArr][ii] =
conArray.get(posArr).getEmpresa().getNombre();
86     } else if (ii == 15) {
87         matrizO[pos][posArr][ii] =
conArray.get(posArr).getListado() + "_n" +
conArray.get(posArr).getReserva();
88     }
89 }
90 }
91 }
92 return matrizO;
93 }

```

Donde es especialmente relevante el primer dato que se obtiene, en la posición 0, se da la coordenada donde se ha posicionado el contenedor, la

fórmula ha de ir recorriendo el buque obteniendo la información de cada contenedor almacenado, para saber su posición recoge su posición dada por un array y la tiene que convertir en el sistema real.

Esta información recopilada se verá en un visor lateral impreso por la clase **Dibujo**.

4.5.2. CLASE DIBUJO

La clase dibujo se encarga de elaborar los planos de carga dado un objeto barco y de crear las leyendas, que son las guías para los códigos de colores.

Los planos de carga reciben un barco y en base a sus dimensiones han de calcular el espacio disponible que tienen para poder realizar los dibujos sin que exista la posibilidad de que queden los gráficos descuadrados.

El ancho del lienzo donde se pintan los barcos es siempre de 1500 píxeles y el alto de 6000. Si el barco es muy grande y el pintado de las bodegas sobrepasa esta longitud, comenzará a trabajar en el nivel siguiente.

Hay dos modos de visualización:


- Modo transversal (

Ilustración 11: Buque Beatriz B en modo transversal. FUENTE: Propia.

Donde los segmentos superior e inferior, las dos bahías simples de una dupla. Obsérvese que la bahía dupla 2 tiene una bahía simple.



Ilustración 12: Bahía dupla 10. Compuesta por dos simples 9 y 11, FUENTE: Propia.

- Modo longitudinal (☞): El barco se pinta de proa a popa mostrando solo los contenedores que están contenidos dentro de una misma fila. Para ello se genera un selector de filas adicional en el panel.

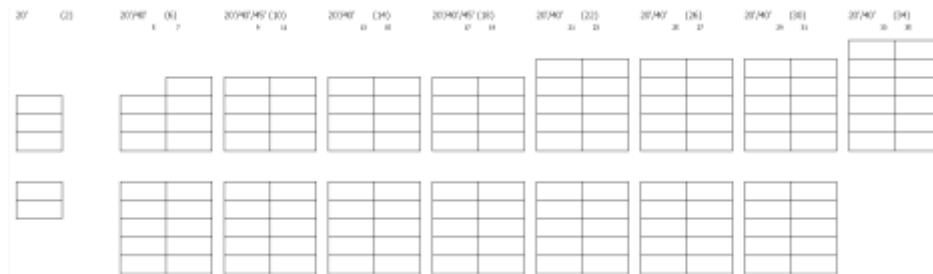


Ilustración 13: Buque Beatriz B en modo longitudinal, fila 1. FUENTE: Propia.

Donde el segmento superior representa la cubierta y el inferior la bodega. La parte de la izquierda la bahía simple de proa y la derecha la de popa.



Ilustración 14: Bahía 10, fila 1. FUENTE: Propia.

Dependiendo del como esté cargado el barco y el color de visualización que escoja el usuario, las casillas cambiarán de color.

Hay los siguientes modos de visualización:

- Puerto de destino (POD). Por defecto. Los datos que la componen dependerán del tipo de ruta que efectúe el barco.
- Puerto de origen (POL). Mismas características que puerto de destino.
- Dimensión.



Ilustración 15: Leyenda de dimensiones. FUENTE: Propia.

- Tipo de contenedor.

■	Contenedor para uso general sin ventilación.
■	Contenedor para uso general ventilado.
■	Contenedor para productos sólidos a granel.
■	Contenedor especializado.
■	Contenedor refrigerado.
■	Contenedor de características térmicas.
■	Contenedor de techo abierto.
■	Contenedor plataforma.
■	Contenedor cisterna.
■	Contenedor especial.

Ilustración 16: Leyenda de tipo. FUENTE: Propia.

- Si el contenedor va vacío.

■	Contenedor con mercancía.
■	Contenedor vacío.

Ilustración 17: Leyenda del MTY. FUENTE: Propia.

- Peso.

■	mas de 25000 kgs.
■	de 20001 hasta 25000 kgs.
■	de 15001 hasta 20000 kgs.
■	de 10001 hasta 15000 kgs.
■	de 5001 hasta 10000 kgs.
■	menos de 5000 kgs.

Ilustración 18: Leyenda de pesajes. FUENTE: Propia.

- Si contiene mercancía peligrosa.

■	1.1: S&O. con riesgo de explosión de toda la n
■	1.2: S&O. con riesgo de explosión de toda la n
■	1.3 S&O. con un riesgo de incendio y un riesg
■	1.4: S&O. sin ningún riesgo considerable.
■	1.5: S. muy insensibles que presentan un ries
■	1.6: O. sumamente insensibles que no presen
■	2.1: Gases inflamables.
■	2.2: Gases no inflamables, no tóxicos.
■	2.3: Gases tóxicos.
■	3: Líquidos inflamables.
■	4.1: Sólidos inflamables, sustancias que pued
■	4.2: S. que pueden experimentar combustión
■	4.3: S. que, en contacto con el agua, desprend
■	5.1: S. comburentes.
■	5.2: Peróxidos orgánicos.
■	6.1: S. tóxicas.
■	6.2: S. infecciosas.
■	7: Material radioactivo.
■	8: S. corrosivas.
■	9: S&O. peligrosos varios.

Ilustración 19: Leyenda de MMPP. FUENTE: Propia.

- Empresa propietaria del contenedor.

Textainer Group Holdings
Triton International Ltd.
A.P. Moller-Maersk Group
Mediterranean Shipping Company S.A.
COSCO
CMA CGM Group
Hapag-Lloyd
Ocean Network Express
Evergreen Marine Corporation
Almar Container Investments Inc.
Tankcon International
Cargostore Worldwide Trading Ltd.
Eurotainer S.A.
Yang Ming Marine Transport Corporation
Protank Logistics CO Ltd.
Thielmann Financial Solutions
Royal Wolf Holdings
VOTG Tanktainer GmbH
Anteng Holdings
Pacific International Lines
Hyundai Merchant Marine
Arkas Line/EMES
ZIM Integrated Shipping Services
Wan Hai Lines
Zhonggu Logistics Corporation

Ilustración 20: Leyenda de empresas, parte 1. FUENTE: Propia.

Islamic Republic of Iran Shipping Lines
Grimaldi
Korea Marine Transport Company
Shandong International Transportation Crp.
TS Lines
Hamburg Sud
SM Line Corporation
Sinotrans
Safmarine Container Lines N.V.
Sinokor Merchant Marine
Regional Container Lines
PT Sinar Pacific Indonesia Lines
Unifeeder
China International Marine Containers Ltd.
The China Navigation Company
Ge-ex Logistics B.V.
AMKON INC
NileDutch
United Arab Shipping
Boluda Lines S.A.
BSL Containers Ltd.
Hacon Containers B.V.
OPDR
Transinsular
Mitsui O.S.K. Lines

Ilustración 21: Leyenda de empresas, parte 2. FUENTE: Propia.

- Tipo de contenedor en base al código (cuatro tipos: contenedor

regular, duplicado para simular los 40'/45', estructural o vacío).





	Espacios vacíos.
	Posición estructural.
	Espacio ocupado por un duplicado.
	Contenedor real.

Ilustración 22: Leyenda de contenedores código. FUENTE: Propia.

Con la salvedad del modo de visualización contenedor en base a código:

- El blanco es el color por defecto que indica la ausencia de contenedor.
- El negro es el color por defecto que indica la incapacidad del programa de leer un contenedor, es una forma fácil de detectar errores.
- El gris se usará en los modos POL y POD para señalar los puertos no incluidos en la ruta.

En la clase dibujo hay unos métodos encargados de imprimir un cuadro de estadísticas con la información recopilada por la clase **Estadísticas**, del capítulo anterior.

Inicio	
Nº de contenedores cargados:	0
Nº de contenedores con MMPP:	0
Nº de espacios vacíos (TEUs):	1267
Peso total de la carga:	0 t.
<hr/>	
nºContenedores / ton en (2)	0 / 0 t.
nºContenedores / ton en (6)	0 / 0 t.
nºContenedores / ton en (10)	0 / 0 t.
nºContenedores / ton en (14)	0 / 0 t.
nºContenedores / ton en (18)	0 / 0 t.
nºContenedores / ton en (22)	0 / 0 t.
nºContenedores / ton en (26)	0 / 0 t.
nºContenedores / ton en (30)	0 / 0 t.
nºContenedores / ton en (34)	0 / 0 t.

Ilustración 23: Cuadro de estadísticas del Beatriz B estando vacío. FUENTE: Propia.

5. ALGORITMOS DE CARGA

Los algoritmos de carga son los métodos encargados de, en base a un barco y una ruta, introducir los contenedores en el barco ateniéndose a una serie de criterios. Se alojan en la clase **Viaje**.

Estos son los métodos más complejos del programa, por ello se comenzó por una primera iteración, `cargaBasica_v0` que introducía contenedores sin requerimientos, y en cada versión posterior se le han añadido funcionalidades.

Todos los algoritmos de carga tienen la siguiente estructura:

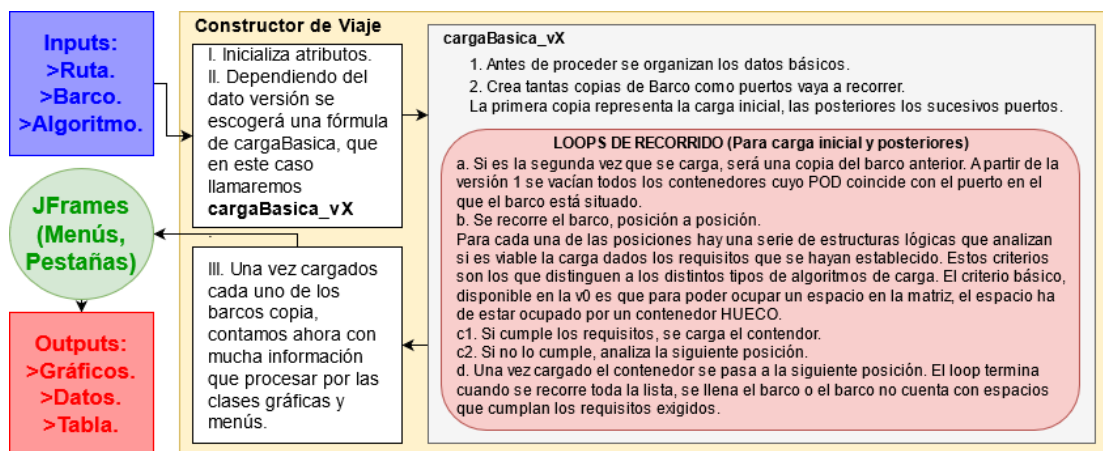


Ilustración 24: Estructura de un algoritmo de carga. FUENTE: Propia.

Esta es la solución presentada, pero no tiene por qué ser la única. Al tratarse de un algoritmo que combina tantas variables, tendrá varias formas con las que se pueda afrontar este problema.

5.1. VERSIONES

En la siguiente tabla se resumen las capacidades de cada algoritmo de carga:

VERSIÓN	Descarga contenedores	Apilamiento en alto	Apilamiento en ancho	Estiba combinada	Puerto de destino	Restricciones tipocon	Pesos	Escotillas	Mmpp
CARGABASICA_V0	NO	NO	NO	NO	NO	NO	NO	NO	NO
CARGABASICA_V1	SÍ	NO	NO	NO	NO	NO	NO	NO	NO
CARGABASICA_V2	SÍ	NO	NO	NO	SÍ	SÍ	SÍ	NO	NO
CARGABASICA_V3	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	NO	NO
CARGABASICA_V4	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	NO
CARGABASICA_V5	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	NO	SÍ
CARGABASICA_V6	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ

En los que cada una de las funciones es la siguiente:

- **Descarga contenedores**: Capacidad del algoritmo para descargar los contenedores según el puerto al que llegue el barco.
- **Apilamientos en alto y en ancho**: Los planos de carga indican que bahías están reservadas a los contenedores según su dimensión. Habrá zonas que estén restringidas a los contenedores de 40'/45', o simplemente no puedan ser cargados por limitaciones de espacio. Un contenedor de altura estándar solo puede ser cargado sobre contenedores de su misma altura, lo mismo ocurre con los high cube.
- **Restricciones tipocon**: Limitación dada por el diseño del buque que indica en que bahías pueden estibarse los contenedores dadas sus dimensiones.
- **Puerto de destino**: Un contenedor no puede ser cargado sobre un contenedor que sale antes de acuerdo con su ruta. Los contenedores solo pueden ser apilados con contenedores de su mismo POD o que el contenedor inferior salga en puertos posteriores. Un contenedor que se posicione en cubierta ha de salir antes que cualquiera de los contenedores que se encuentran en la bodega de su bahía, puesto que los contenedores de cubierta se posicionan sobre las escotillas.
- **Pesos**: Los contenedores más pesados han de ir por debajo de los más ligeros. Preferiblemente en bodega. De esta manera evitamos que el centro de gravedad del buque ascienda más de lo conveniente.
- **Visibilidad**: Los contenedores no pueden suponer una intrusión para

la visibilidad de puente. Puesto que la altura máxima en la que se apilan viene dada por las restricciones de la matriz este criterio viene por defecto en cualquier algoritmo de carga que se haga.

- **Estiba combinada**: Con contenedores de 20' y de 40'. Un contenedor de 40' puede ser cargado sobre dos contenedores de 20' pero un 20' no puede ir sobre un contenedor de 40'. Esto se le conoce como estiba combinada, o rusa.
- **Presencia de mercancía IMO**: Que tendrá que atenerse a las instrucciones del código IMDG.

Como puede observarse, el algoritmo `cargaBasica_v0` no hace nada aparte de cargar contenedores. Sin embargo, al ser tan sencilla conviene explicar como funciona para comprender como se estructuran las otras funciones.

```
1 public Barco[] cargaBasica_v0() {
2     //Listas de la carga.
3     this.cargas = new Barco[this.getRuta().getPuertos().size() -
4     1]; //Crea un array de barcos que simulan las distintas cargas que habrá a lo
5     largo del viaje, una por puerto visitado.
6     this.getBarco().guardarBarco(); //El barco modelo se utilizará para
7     cargar primero la lista inicial.
8     this.cargas[0] =
9     this.getBarco().cargarBarco(this.getBarco().getNombre()); //La primera
10    posición del barco se llena con el barco previo al primer puerto. Se
11    modificará si hay una cargaPrevia.
12    for (int pos = 0; pos < this.cargas.length; pos++) { //Contador de
13    las posiciones del barco a lo largo del viaje.
14        if (pos > 0) { //Se cambia el barco al ir a un puerto nuevo
15            this.cargas[pos - 1].guardarBarco();
16            this.cargas[pos] =
17            this.getBarco().cargarBarco(this.getBarco().getNombre()); //El barco de
18            llegada al nuevo puerto tendrá la misma carga que el barco que salió del
19            puerto anterior.
20        }
21        int sum = 0; //Suma el contador de elementos de la lista,
22        cuando se iguala al n° de elementos de la lista lanza un break para ponerse
23        con el siguiente barco.
24        for (int i = 0; i < this.getBarco().getMatriz().length; i++) {
25            for (int j = 0; j < this.getBarco().getMatriz()[i].length;
26            j++) {
27                if (this.getBarco().getMatriz()[i][j] != null) {
28                    for (int k = 0; k <
29                    this.getBarco().getMatriz()[i][j].length; k++) {
30                        if (this.getBarco().getMatriz()[i][j][k] !=
31                        null) {
32                            for (int m = 0; m <
33                            this.getBarco().getMatriz()[i][j][k].length; m++) {
34                                for (int n = 0; n <
35                                this.getBarco().getMatriz()[i][j][k][m].length; n++) { //Se llega al nivel
```

```

    mas bajo del barco, comienza la sustitución.
19         if (this.getRuta().getCargaPrevia()
    != null && pos == 0) {//La posición 0 se reserva para el barco previo a las
    rutas, puede rellenarse si hay una lista de carga previa.
20             if
    (this.getBarco().getMatriz()[i][j][k][m][n].getId().equals("HUECO")) {
21                 if (sum <
    this.getRuta().getCargaPrevia().length) {//Mientras no se rebase la longitud
    de la lista.
22
    this.cargas[pos].getMatriz()[i][j][k][m][n] =
    this.getRuta().getCargaPrevia()[sum];
23
    sum = sum + 1;
24             } else {
25                 break;
26             }
27         }
28         } else if (pos > 0) {//Rellenos para
    las posiciones distintas a la inicial.
29             if
    (this.cargas[pos].getMatriz()[i][j][k][m][n].getId().equals("HUECO")) {
30                 if (sum <
    this.getRuta().getListaDeListas().get(pos - 1).length) {//Mientras no se
    rebase la longitud de la lista.
31
    this.cargas[pos].getMatriz()[i][j][k][m][n]
    = this.getRuta().getListaDeListas().get(pos - 1)[sum];
32
    sum = sum + 1;
33             } else {
34                 break;
35             }
36         }
37     }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46     return this.cargas;
47 }

```

Para futuras iteraciones de esta fórmula, nos vamos a ayudar de un método llamado ley de apilamiento, que estudia el contenido de un contenedor y el que tiene abajo para ver si puede posicionarlo encima. En la versión v3 devolverá un valor lógico (verdadero o falso), dependiendo de su peso, su altura y su POD.

```

1 public boolean leyDeApilamento_v3(int pos, Ruta ruta, Contenedor
    posInferior, Contenedor con) {
2     Boolean res = false;
3     if (con.getVgm() < posInferior.getVgm() && (con.getDyt().charAt(1)
    == posInferior.getDyt().charAt(1))) {//Compara los pesos y alturas

```

```

4         try {
5             if
        (con.getPod().getLocode().equals(posInferior.getPod().getLocode())) {//Si
        salen en el mismo puerto
6                 res = true;
7             } else //En el caso de que no haya una igualdad
8                 ArrayList<Puerto> arrP = new ArrayList<Puerto>();
9                 for (int i = pos; ruta.getPuertos().size() > i; i++) {
10                     arrP.add(ruta.getPuertos().get(i));
11                 }
12                 int NCon = 0;
13                 int NposInferior = 0;
14                 lectura:
15                 for (int i = 0; arrP.size() > i; i++) {
16                     if
        (arrP.get(i).getLocode().equals(con.getPod().getLocode())) {
17                         NCon = i;
18                         break lectura;
19                     }
20                 }
21                 lectura2:
22                 for (int i = 0; arrP.size() > i; i++) {
23                     if
        (arrP.get(i).getLocode().equals(posInferior.getPod().getLocode())) {
24                         NposInferior = i;
25                         break lectura2;
26                     }
27                 }
28                 if (NCon < NposInferior) {
29                     res = true;
30                 }
31             }
32         } catch (Exception e) {
33             return res;
34         }
35     }
36     return res;
37 }

```

El siguiente método es `cargaBásica_v3`, que se basa en el mismo funcionamiento que el algoritmo `v0` anterior con la diferencia que va estudiando su posición en el barco con respecto las posiciones entre el contenedor a introducir y su entorno, apoyo que en parte le da. Para ello cuenta con varias estructuras lógicas adicionales y se apoya en la función `leyDeApilamiento_v3` vista antes.

```

1 public Barco[] cargaBasica_v3() {
2     //Listas de la carga.
3     this.cargas = new Barco[this.getRuta().getPuertos().size() -
        1];//Crea un array de barcos que simulan las distintas cargas que habrá a
        lo largo del viaje, una por puerto visitado.
4     this.getBarco().guardarBarco();//El barco modelo se utilizará para
        cargar primero la lista inicial.

```

```

5         this.cargas[0] =
        this.getBarco().cargarBarco(this.getBarco().getNombre()); //La primera
        posición del barco se llena con el barco previo al primer puerto. Se
        modificará si hay una cargaPrevia.
6         for (int pos = 0; pos < this.cargas.length; pos++) { //Contador de
        las posiciones del barco a lo largo del viaje.
7             if (pos > 0) { //Se cambia el barco al ir a un puerto nuevo
8                 this.cargas[pos - 1].guardarBarco();
9                 this.cargas[pos] =
        this.getBarco().cargarBarco(this.getBarco().getNombre()); //El barco de
        llegada al nuevo puerto tendrá la misma carga que el barco que salió del
        puerto anterior.
10                this.descargaBarco(pos); //QUITAR CONTENEDORES
11            }
12            //AÑADIR CONTENEDORES
13            int sum = 0; //Suma el el contador de elementos de la lista,
        cuando se iguala al n° de elementos de la lista lanza un break para ponerse
        con el siguiente barco.
14            int contador = 0;
15            if (pos == 0 && this.getRuta().getCargaPrevia() == null) {
16                contador = sum;
17            } else if (pos == 0) {
18                contador = this.getRuta().getCargaPrevia().length;
19            } else {
20                contador = this.getRuta().getListaDeListas().get(pos -
        1).length;
21            }
22            int loops = 0;
23            while (sum < contador) {
24                barco:
25                for (int i = 0; i < this.getBarco().getMatriz().length;
        i++) {
26                    for (int j = 0; j <
        this.getBarco().getMatriz()[i].length; j++) {
27                        if (this.getBarco().getMatriz()[i][j] != null) {
28                            for (int k = 0; k <
        this.getBarco().getMatriz()[i][j].length; k++) {
29                                if (this.getBarco().getMatriz()[i][j][k] !=
        null) {
30                                    for (int m = 0; m <
        this.getBarco().getMatriz()[i][j][k].length; m++) {
31                                        for (int n = 0; n <
        this.getBarco().getMatriz()[i][j][k][m].length; n++) { //Se llega al nivel
        mas bajo del barco, comienza la sustitución.
32                                            if (pos == 0) { //La posición 0
        se reserva para el barco previo a las rutas, puede rellenarse si hay una
        lista de carga previa.
33                                                if
        (this.cargas[pos].getMatriz()[i][j][k][m][n].getId().equals("HUECO") //La
        posición seleccionada es libre
34                                                    &&
        this.getBarco().tipoConComprobador(i,
        this.getRuta().getCargaPrevia()[sum])) { //La posición se encuentra en una
        bahía que permite contenedores de su dimensión
35                                                    if
        (this.getRuta().getCargaPrevia()[sum].dyt.charAt(0) == '2') {
36                                                        if (k == 0) { //Si

```

```

    es en bodega
37                                     if (n == 0 &&
    this.cargas[pos].getMatriz()[i][j][k][m][n].getId().equals("HUECO") &&
    !this.cargas[pos].getMatriz()[i][j][k][m][n].getId().equals("ESTRUCTURAL"))
38 {
39 this.cargas[pos].getMatriz()[i][j][k][m][n] =
    this.getRuta().getCargaPrevia()[sum];
40                                     sum = sum +
    1;
41                                     break
    barco;
42                                     } else if
    (!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO")
    && ((this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getDyt().charAt(0) ==
    '2') && (this.leyDeApilamento_v3(pos, this.getRuta(),
    this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
    this.getRuta().getCargaPrevia()[sum])) ||
    (this.cargas[pos].getMatriz()[i][j][k][m][n -
    1].getId().equals("ESTRUCTURAL")))) {
43 this.cargas[pos].getMatriz()[i][j][k][m][n] =
    this.getRuta().getCargaPrevia()[sum];
44                                     sum = sum +
    1;
45                                     break
    barco;
46                                     }
47                                     } else { //Si es en
    cubierta
48                                     if (n == 0) {
49 this.cargas[pos].getMatriz()[i][j][k][m][n] =
    this.getRuta().getCargaPrevia()[sum];
50                                     sum = sum +
    1;
51                                     break
    barco;
52                                     } else if
    ((!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO"))
    && ((this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getDyt().charAt(0) ==
    '2') && (this.leyDeApilamento_v3(pos, this.getRuta(),
    this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
    this.getRuta().getCargaPrevia()[sum])) ||
    (this.cargas[pos].getMatriz()[i][j][k][m][n -
    1].getId().equals("ESTRUCTURAL")))) {
53 this.cargas[pos].getMatriz()[i][j][k][m][n] =
54 this.getRuta().getCargaPrevia()[sum];
55                                     sum = sum +
    1;
56                                     break
57 barco;
58                                     }
59                                     }
60                                     } else if
    (this.getRuta().getCargaPrevia()[sum].dyt.charAt(0) == '4' ||
    this.getRuta().getCargaPrevia()[sum].dyt.charAt(0) == 'L') {
61                                     if (k == 0) { //Si
    es en bodega
62                                     if

```

```

        (this.cargas[pos].getMatriz()[i][1] == null) {//Es en una bahía única
63                                     if
        ((this.cargas[pos].getMatriz()[i][j][k][m].length - 1) - n) ==
        (this.cargas[pos].getMatriz()[i][j][k][m].length - 1)) {
64 this.cargas[pos].getMatriz()[i][j][k][m][n] =
65 this.getRuta().getCargaPrevia()[sum];
66                                     sum =
        sum + 1;
67                                     break
        barco;
68                                     } else if
        (!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO")
        && ((this.leyDeApilamento_v3(pos, this.getRuta(),
        this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
        this.getRuta().getCargaPrevia()[sum])) ||
        (this.cargas[pos].getMatriz()[i][j][k][m][n -
        1].getId().equals("ESTRUCTURAL")))) {
69 this.cargas[pos].getMatriz()[i][j][k][m][n] =
        this.getRuta().getCargaPrevia()[sum];
70                                     sum =
        sum + 1;
71                                     break
        barco;
72                                     }
73                                     } else {//Es
        una bahía dupla
74                                     if (j == 0
        && ((this.cargas[pos].getMatriz()[i][j][k][m].length - 1) - n) ==
        (this.cargas[pos].getMatriz()[i][j][k][m].length - 1)) {
75                                     if
        (this.cargas[pos].getMatriz()[i][1][k][m][n].getId().equals("HUECO")) {
76 this.cargas[pos].getMatriz()[i][0][k][m][n] =
77 this.getRuta().getCargaPrevia()[sum];
78 this.cargas[pos].getMatriz()[i][1][k][m][n] = new
        Contenedor(this.getRuta().getCargaPrevia()[sum].getListado(),
79 this.getRuta().getCargaPrevia()[sum].getReserva(), "DUPLICADO",
80 this.getRuta().getCargaPrevia()[sum].getDyt(),
81 this.getRuta().getCargaPrevia()[sum].getPol(),
82 this.getRuta().getCargaPrevia()[sum].getPod(),
83 this.getRuta().getCargaPrevia()[sum].getPesoTara(),
84 this.getRuta().getCargaPrevia()[sum].getPesoUtil(),
85 this.getRuta().getCargaPrevia()[sum].pesoCarga,
86 this.getRuta().getCargaPrevia()[sum].getVgm(),
87 this.getRuta().getCargaPrevia()[sum].getMmpp(),
88 this.getRuta().getCargaPrevia()[sum].getEmpresa());
89                                     sum
        = sum + 1;
90 break barco;
91                                     }
92                                     } else if
        (j == 0 && !this.cargas[pos].getMatriz()[i][j][k][m][n -
        1].getId().equals("HUECO") && ((this.leyDeApilamento_v3(pos,
        this.getRuta(), this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
        this.getRuta().getCargaPrevia()[sum])) ||
        (this.cargas[pos].getMatriz()[i][j][k][m][n -
        1].getId().equals("ESTRUCTURAL")) && ((this.leyDeApilamento_v3(pos,
        this.getRuta(), this.cargas[pos].getMatriz()[i][1][k][m][n - 1],

```

```

    this.getRuta().getCargaPrevia()[sum])) ||
    (this.cargas[pos].getMatriz()[i][1][k][m][n -
1].getId().equals("ESTRUCTURAL")))) {
93                                     if
    (!this.cargas[pos].getMatriz()[i][1][k][m][n - 1].getId().equals("HUECO"))
    {
94 this.cargas[pos].getMatriz()[i][0][k][m][n] =
95 this.getRuta().getCargaPrevia()[sum];
96 this.cargas[pos].getMatriz()[i][1][k][m][n] = new
    Contenedor(this.getRuta().getCargaPrevia()[sum].getListado(),
97 this.getRuta().getCargaPrevia()[sum].getReserva(), "DUPLICADO",
98 this.getRuta().getCargaPrevia()[sum].getDyt(),
99 this.getRuta().getCargaPrevia()[sum].getPol(),
100 this.getRuta().getCargaPrevia()[sum].getPod(),
101 this.getRuta().getCargaPrevia()[sum].getPesoTara(),
102 this.getRuta().getCargaPrevia()[sum].getPesoUtil(),
103 this.getRuta().getCargaPrevia()[sum].pesoCarga,
104 this.getRuta().getCargaPrevia()[sum].getVgm(),
105 this.getRuta().getCargaPrevia()[sum].getMmpp(),
106 this.getRuta().getCargaPrevia()[sum].getEmpresa());
107                                     sum
    = sum + 1;
108 break barco;
109                                     }
110                                     }
111                                     }
112                                     } else { //Si es en
    cubierta
113                                     if
    (this.cargas[pos].getMatriz()[i][1] == null) { //Es en una bahía única
114                                     if (n == 0)
115 {
116 this.cargas[pos].getMatriz()[i][j][k][m][n] =
117 this.getRuta().getCargaPrevia()[sum];
118                                     sum =
    sum + 1;
119                                     break
    barco;
120                                     } else if
    ((!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO"))
    && ((this.leyDeApilamento_v3(pos, this.getRuta(),
    this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
    this.getRuta().getCargaPrevia()[sum])) ||
    (this.cargas[pos].getMatriz()[i][j][k][m][n -
    1].getId().equals("ESTRUCTURAL")))) {
121 this.cargas[pos].getMatriz()[i][j][k][m][n] =
122 this.getRuta().getCargaPrevia()[sum];
123                                     sum =
    sum + 1;
124                                     break
    barco;
125                                     }
126                                     } else { //Es
    una bahía dupla
127                                     if (j == 0
    && n == 0) {
128                                     if

```



```

        (this.cargas[pos].getMatriz()[i][1][k][m][n].getId().equals("HUECO")) {
129 this.cargas[pos].getMatriz()[i][0][k][m][n] =
130 this.getRuta().getCargaPrevia()[sum];
131 this.cargas[pos].getMatriz()[i][1][k][m][n] = new
    Contenedor(this.getRuta().getCargaPrevia()[sum].getListado(),
132 this.getRuta().getCargaPrevia()[sum].getReserva(), "DUPLICADO",
133 this.getRuta().getCargaPrevia()[sum].getDyt(),
134 this.getRuta().getCargaPrevia()[sum].getPol(),
135 this.getRuta().getCargaPrevia()[sum].getPod(),
136 this.getRuta().getCargaPrevia()[sum].getPesoTara(),
137 this.getRuta().getCargaPrevia()[sum].getPesoUtil(),
138 this.getRuta().getCargaPrevia()[sum].pesoCarga,
139 this.getRuta().getCargaPrevia()[sum].getVgm(),
140 this.getRuta().getCargaPrevia()[sum].getMmpp(),
141 this.getRuta().getCargaPrevia()[sum].getEmpresa());
142
sum
    = sum + 1;
143 break barco;
144
145
        } else if
        (j == 0 && (!this.cargas[pos].getMatriz()[i][j][k][m][n -
1].getId().equals("HUECO")) && ((this.leyDeApilamento_v3(pos,
this.getRuta(), this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
this.getRuta().getCargaPrevia()[sum])) ||
(this.cargas[pos].getMatriz()[i][j][k][m][n -
1].getId().equals("ESTRUCTURAL")) && ((this.leyDeApilamento_v3(pos,
this.getRuta(), this.cargas[pos].getMatriz()[i][1][k][m][n - 1],
this.getRuta().getCargaPrevia()[sum])) ||
(this.cargas[pos].getMatriz()[i][1][k][m][n -
1].getId().equals("ESTRUCTURAL")))) {
146
if
        (!this.cargas[pos].getMatriz()[i][1][k][m][n - 1].getId().equals("HUECO"))
147 {
148 this.cargas[pos].getMatriz()[i][0][k][m][n] =
149 this.getRuta().getCargaPrevia()[sum];
150 this.cargas[pos].getMatriz()[i][1][k][m][n] = new
    Contenedor(this.getRuta().getCargaPrevia()[sum].getListado(),
151 this.getRuta().getCargaPrevia()[sum].getReserva(), "DUPLICADO",
152 this.getRuta().getCargaPrevia()[sum].getDyt(),
153 this.getRuta().getCargaPrevia()[sum].getPol(),
154 this.getRuta().getCargaPrevia()[sum].getPod(),
155 this.getRuta().getCargaPrevia()[sum].getPesoTara(),
156 this.getRuta().getCargaPrevia()[sum].getPesoUtil(),
157 this.getRuta().getCargaPrevia()[sum].pesoCarga,
158 this.getRuta().getCargaPrevia()[sum].getVgm(),
159 this.getRuta().getCargaPrevia()[sum].getMmpp(),
160 this.getRuta().getCargaPrevia()[sum].getEmpresa());
161
sum
    = sum + 1;
162 break barco;
163
164
165
166
167
168
169
        } else if (pos > 0) { //Rellenos

```

```

    para las posiciones distintas a la inicial.
170                                     if
171 (this.cargas[pos].getMatriz()[i][j][k][m][n].getId().equals("HUECO"))//La
    posición seleccionada es libre
172                                     &&
173 this.getBarco().tipoConComprobador(i,
174 this.getRuta().getListaDeListas().get(pos - 1)[sum])) {//La posición se
    encuentra en una bahía que permite contenedores de su dimensión
175                                     if
176 (this.getRuta().getListaDeListas().get(pos - 1)[sum].dyt.charAt(0) == '2')
177 {
178                                     if (k == 0) {//Si
    es en bodega
179                                     if (n == 0 &&
    this.cargas[pos].getMatriz()[i][j][k][m][n].getId().equals("HUECO") &&
    !this.cargas[pos].getMatriz()[i][j][k][m][n].getId().equals("ESTRUCTURAL"))
180 {
181     this.cargas[pos].getMatriz()[i][j][k][m][n] =
182     this.getRuta().getListaDeListas().get(pos - 1)[sum];
183                                     sum = sum +
    1;
184                                     break
    barco;
185                                     } else if
186 (!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO")
    && ((this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getDyt().charAt(0) ==
    '2') && (this.leyDeApilamento_v3(pos, this.getRuta(),
    this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
    this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||
    (this.cargas[pos].getMatriz()[i][j][k][m][n -
    1].getId().equals("ESTRUCTURAL")))) {
187 this.cargas[pos].getMatriz()[i][j][k][m][n] =
188 this.getRuta().getListaDeListas().get(pos - 1)[sum];
189                                     sum = sum +
    1;
190                                     break
    barco;
191                                     }
192                                     } else {//Si es en
    cubierta
193                                     if (n == 0) {
194 this.cargas[pos].getMatriz()[i][j][k][m][n] =
195 this.getRuta().getListaDeListas().get(pos - 1)[sum];
196                                     sum = sum +
    1;
197                                     break
    barco;
198                                     } else if
199 ((!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO"))
    && ((this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getDyt().charAt(0) ==
    '2') && (this.leyDeApilamento_v3(pos, this.getRuta(),
    this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
    this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||
    (this.cargas[pos].getMatriz()[i][j][k][m][n -
    1].getId().equals("ESTRUCTURAL")))) {
200 this.cargas[pos].getMatriz()[i][j][k][m][n] =
201 this.getRuta().getListaDeListas().get(pos - 1)[sum];

```

```

195 sum = sum +
    1;
196 break
    barco;
197 }
198 }
199 } else if
    (this.getRuta().getListaDeListas().get(pos - 1)[sum].dyt.charAt(0) == '4'
    || this.getRuta().getListaDeListas().get(pos - 1)[sum].dyt.charAt(0) ==
    'L') {
200 if (k == 0) { //Si
    es en bodega
201 if
    (this.cargas[pos].getMatriz()[i][1] == null) { //Es en una bahía única
202 if
    (((this.cargas[pos].getMatriz()[i][j][k][m].length - 1) - n) ==
    (this.cargas[pos].getMatriz()[i][j][k][m].length - 1)) {
203 this.cargas[pos].getMatriz()[i][j][k][m][n] =
204 this.getRuta().getListaDeListas().get(pos - 1)[sum];
205 sum =
    sum + 1;
206 break
    barco;
207 } else if
    (!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO")
    && ((this.leyDeApilamento_v3(pos, this.getRuta(),
    this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
    this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||
    (this.cargas[pos].getMatriz()[i][j][k][m][n -
    1].getId().equals("ESTRUCTURAL")))) {
208 this.cargas[pos].getMatriz()[i][j][k][m][n] =
209 this.getRuta().getListaDeListas().get(pos - 1)[sum];
210 sum =
    sum + 1;
211 break
    barco;
212 }
213 } else { //Es
    una bahía dupla
214 if (j == 0
    && ((this.cargas[pos].getMatriz()[i][j][k][m].length - 1) - n) ==
215 (this.cargas[pos].getMatriz()[i][j][k][m].length - 1)) {
    (this.cargas[pos].getMatriz()[i][1][k][m][n].getId().equals("HUECO")) {
216 this.cargas[pos].getMatriz()[i][0][k][m][n] =
    this.getRuta().getListaDeListas().get(pos - 1)[sum];
217 this.cargas[pos].getMatriz()[i][1][k][m][n] = new
    Contenedor(this.getRuta().getListaDeListas().get(pos -
    1)[sum].getListado(), this.getRuta().getListaDeListas().get(pos -
    1)[sum].getReserva(), "DUPLICADO",
218 this.getRuta().getListaDeListas().get(pos - 1)[sum].getDyt(),
219 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPol(),
220 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPod(),
221 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoTara(),
222 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoUtil(),
223 this.getRuta().getListaDeListas().get(pos - 1)[sum].pesoCarga,
224 this.getRuta().getListaDeListas().get(pos - 1)[sum].getVgm(),
225 this.getRuta().getListaDeListas().get(pos - 1)[sum].getMmpp(),

```

```

226 this.getRuta().getListaDeListas().get(pos - 1)[sum].getEmpresa());
227                                                                 sum
228 = sum + 1;
229 break barco;
230                                                                 }
231                                                                 } else if
    (j == 0 && !this.cargas[pos].getMatriz()[i][j][k][m][n -
1].getId().equals("HUECO") && ((this.leyDeApilamento_v3(pos,
this.getRuta(), this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||
(this.cargas[pos].getMatriz()[i][j][k][m][n -
1].getId().equals("ESTRUCTURAL"))) && ((this.leyDeApilamento_v3(pos,
this.getRuta(), this.cargas[pos].getMatriz()[i][1][k][m][n - 1],
this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||
(this.cargas[pos].getMatriz()[i][1][k][m][n -
1].getId().equals("ESTRUCTURAL")))) {
232                                                                 if
    (!this.cargas[pos].getMatriz()[i][1][k][m][n - 1].getId().equals("HUECO"))
233 {
234 this.cargas[pos].getMatriz()[i][0][k][m][n] =
235 this.getRuta().getListaDeListas().get(pos - 1)[sum];
236 this.cargas[pos].getMatriz()[i][1][k][m][n] = new
    Contenedor(this.getRuta().getListaDeListas().get(pos -
1)[sum].getListado(), this.getRuta().getListaDeListas().get(pos -
1)[sum].getReserva(), "DUPLICADO",
237 this.getRuta().getListaDeListas().get(pos - 1)[sum].getDyt(),
238 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPol(),
239 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPod(),
240 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoTara(),
241 this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoUtil(),
242 this.getRuta().getListaDeListas().get(pos - 1)[sum].pesoCarga,
243 this.getRuta().getListaDeListas().get(pos - 1)[sum].getVgm(),
244 this.getRuta().getListaDeListas().get(pos - 1)[sum].getMmpp(),
245 this.getRuta().getListaDeListas().get(pos - 1)[sum].getEmpresa());
246                                                                 sum
    = sum + 1;
247 break barco;
248                                                                 }
249                                                                 }
250                                                                 }
251                                                                 } else { //Si es en
    cubierta
252                                                                 if
    (this.cargas[pos].getMatriz()[i][1] == null) { //Es en una bahía única
253                                                                 if (n == 0)
    {
254 this.cargas[pos].getMatriz()[i][j][k][m][n] =
255 this.getRuta().getListaDeListas().get(pos - 1)[sum];
256                                                                 sum =
    sum + 1;
257                                                                 break
    barco;
258                                                                 } else if
    ((!this.cargas[pos].getMatriz()[i][j][k][m][n - 1].getId().equals("HUECO"))
&& ((this.leyDeApilamento_v3(pos, this.getRuta(),
this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||

```

```

        (this.cargas[pos].getMatriz()[i][j][k][m][n -
1].getId().equals("ESTRUCTURAL")))) {
259 this.cargas[pos].getMatriz()[i][j][k][m][n] =
260 this.getRuta().getListaDeListas().get(pos - 1)[sum];
261
sum =
    sum + 1;
262
break
    barco;
263
}
264
} else { //Es
    una bahía dupla
265
if (j == 0
    && n == 0) {
266
if
        (this.cargas[pos].getMatriz()[i][1][k][m][n].getId().equals("HUECO")) {
        this.cargas[pos].getMatriz()[i][0][k][m][n] =
        this.getRuta().getListaDeListas().get(pos - 1)[sum];
        this.cargas[pos].getMatriz()[i][1][k][m][n] = new
        Contenedor(this.getRuta().getListaDeListas().get(pos -
1)[sum].getListado(), this.getRuta().getListaDeListas().get(pos -
1)[sum].getReserva(), "DUPLICADO",
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getDyt(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getPol(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getPod(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoTara(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoUtil(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].pesoCarga,
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getVgm(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getMmpp(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getEmpresa());
267
sum
        = sum + 1;
268 break barco;
269
}
270
} else if
        (j == 0 && (!this.cargas[pos].getMatriz()[i][j][k][m][n -
1].getId().equals("HUECO")) && ((this.leyDeApilamento_v3(pos,
        this.getRuta(), this.cargas[pos].getMatriz()[i][j][k][m][n - 1],
        this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||
        (this.cargas[pos].getMatriz()[i][j][k][m][n -
1].getId().equals("ESTRUCTURAL")) && ((this.leyDeApilamento_v3(pos,
        this.getRuta(), this.cargas[pos].getMatriz()[i][1][k][m][n - 1],
        this.getRuta().getListaDeListas().get(pos - 1)[sum])) ||
        (this.cargas[pos].getMatriz()[i][1][k][m][n -
1].getId().equals("ESTRUCTURAL")))) {
271
if
        (!this.cargas[pos].getMatriz()[i][1][k][m][n - 1].getId().equals("HUECO"))
272 {
273 this.cargas[pos].getMatriz()[i][0][k][m][n] =
274 this.getRuta().getListaDeListas().get(pos - 1)[sum];
275 this.cargas[pos].getMatriz()[i][1][k][m][n] = new
        Contenedor(this.getRuta().getListaDeListas().get(pos -
1)[sum].getListado(), this.getRuta().getListaDeListas().get(pos -
1)[sum].getReserva(), "DUPLICADO",
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getDyt(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getPol(),
        this.getRuta().getListaDeListas().get(pos - 1)[sum].getPod(),

```

```

    this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoTara(),
    this.getRuta().getListaDeListas().get(pos - 1)[sum].getPesoUtil(),
    this.getRuta().getListaDeListas().get(pos - 1)[sum].pesoCarga,
    this.getRuta().getListaDeListas().get(pos - 1)[sum].getVgm(),
    this.getRuta().getListaDeListas().get(pos - 1)[sum].getMmpp(),
    this.getRuta().getListaDeListas().get(pos - 1)[sum].getEmpresa());
276                                     sum
    = sum + 1;
277 break barco;
278                                     }
279                                     }
280                                     }
281                                     }
282                                     }
283                                     }
284                                     }
285                                     }
286                                     }
287                                     }
288                                     }
289                                     }
290                                     }
291                                     }
292         sum = sum + 1;
293         loops = loops + 1;
294     }
295     System.out.println("Total de loops: " + loops);
296 }
297 return this.cargas;
298 }

```

5.2. SEGREGACIÓN DE MMPP

Este es el método encargado de la segregación en la clase viaje devolverá la posibilidad de cargar un contenedor de acuerdo con el tipo de mercancía peligrosa que lleve. En su elaboración se ha basado en el diagrama adjunto del apartado 7.2 código IMDG.

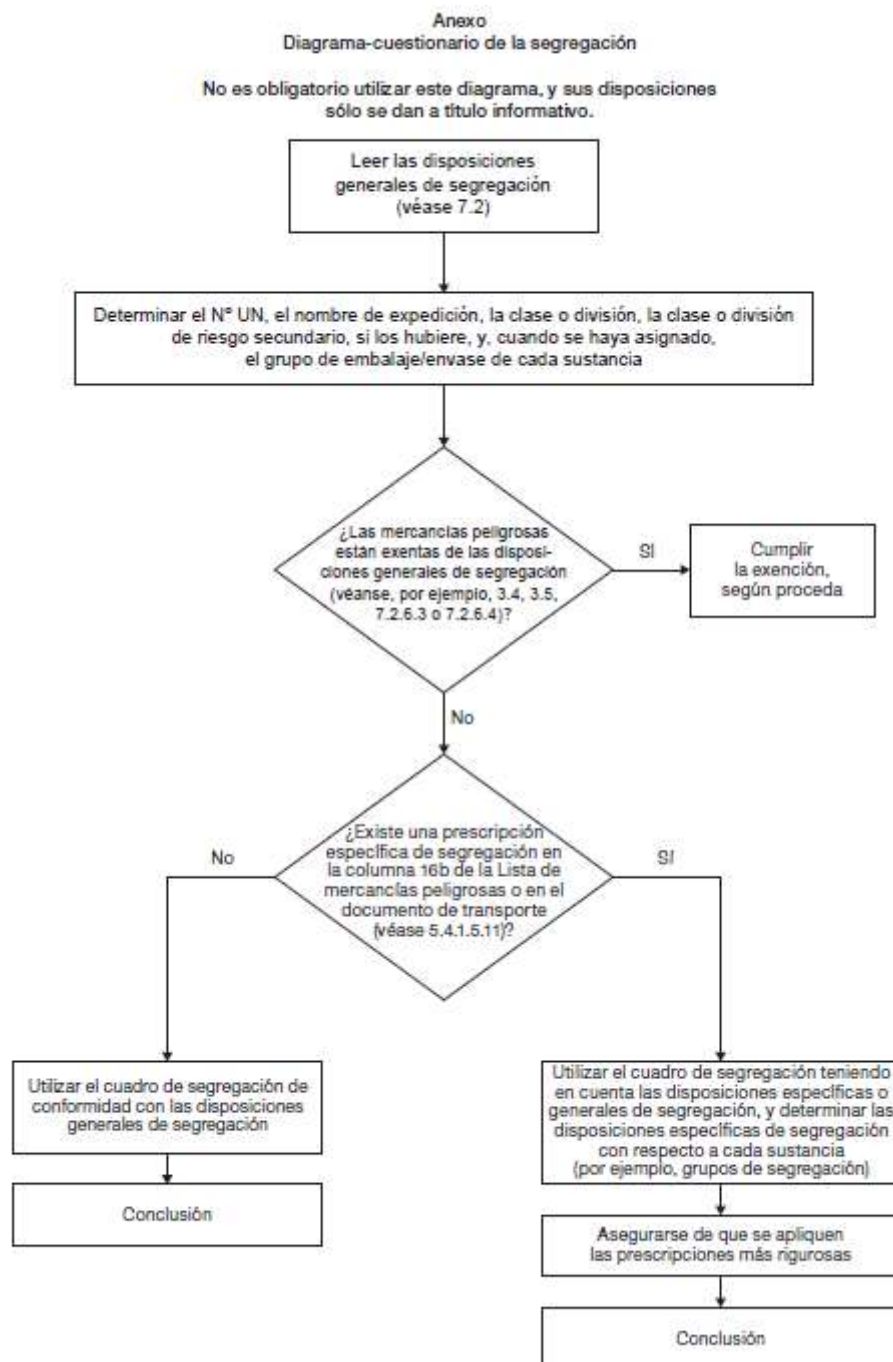


Ilustración 25: Diagrama-cuestionario de segregación. FUENTE: Código IMDG.

Considerando lo siguiente:

- Los buques son con escotillas estancas y bodegas ventiladas.
- La disposición 7.2.6.3 en la cual se detalla que no es necesario realizar la segregación de dos sustancias pese a tener distinta clase si son la misma sustancia, pero en distintas concentraciones o disoluciones. No hay atributo en la clase **Mmpp** que indique que son misma sustancia, para emular esto se analiza el contenido de los nombres de las mercancías y si encuentra una correlación entre ambas cadenas de texto de más del 90%, se considerará que son misma sustancia.
- Esta es una función de segregación simple, no va a tener en consideración disposiciones específicas, 16b.
- Consideramos siempre un entorno cerrado/cerrado para la hora de seleccionar los criterios de distanciamiento dados por la siguiente tabla:

Segregación exigida	Vertical				Horizontal					
	Cerrado/cerrado	Cerrado/abierto	Abierto/abierto		Cerrado/cerrado		Cerrado/abierto		Abierto/abierto	
					En cubierta	Bajo cubierta	En cubierta	Bajo cubierta	En cubierta	Bajo cubierta
«A distancia de- .1	Permitido uno encima de otro	Permitido abierto sobre cerrado	Prohibido en la misma línea vertical a menos que estén separados por una cubierta	En sentido longitudinal	No hay restricción	No hay restricción	No hay restricción	No hay restricción	Un espacio para contenedor	Un espacio para contenedor o un mamparo
		Si no, igual que para «abierto/abierto»		En sentido transversal	No hay restricción	No hay restricción	No hay restricción	No hay restricción	Un espacio para contenedor	Un espacio para contenedor
«Separado de- .2	Prohibido en la misma línea vertical a menos que estén separados por una cubierta	En sentido longitudinal		Un espacio para contenedor	Un espacio para contenedor o un mamparo	Un espacio para contenedor	Un espacio para contenedor o un mamparo	Un espacio para contenedor	Un mamparo	
		En sentido transversal		Un espacio para contenedor	Un espacio para contenedor	Un espacio para contenedor	Dos espacios para contenedor	Dos espacios para contenedor	Un mamparo	
«Separado por todo un compartimento o toda una bodega de- .3		En sentido longitudinal		Un espacio para contenedor	Un mamparo	Un espacio para contenedor	Un mamparo	Dos espacios para contenedor	Dos mamparos	
		En sentido transversal		Dos espacios para contenedor	Un mamparo	Dos espacios para contenedor	Un mamparo	Tres espacios para contenedor	Dos mamparos	
«Separado longitudinalmente por todo un compartimento intermedio o toda una bodega intermedia de- .4		Prohibido	En sentido longitudinal	Distancia de 24 m por lo menos en sentido horizontal	Un mamparo y distancia de 24 m por lo menos en sentido horizontal	Distancia de 24 m por lo menos en sentido horizontal	Dos mamparos	Distancia de 24 m por lo menos en sentido horizontal	Dos mamparos	
			En sentido transversal	Prohibido	Prohibido	Prohibido	Prohibido	Prohibido	Prohibido	

* Los contenedores a no menos de 8 del mamparo intermedio.

Nota: Todos los mamparos y cubiertas serán resistentes al fuego y a los líquidos.

Ilustración 26: Cuadro de segregación, apartado 7.4.3.2 FUENTE: Código IMDG

Una vez tenidas en cuenta estas consideraciones, se adjunta la fórmula de segregación, `criterioMmpp`, en el que dado un contenedor, sus coordenadas en array y un barco, devolverá un verdadero si se puede cargar de acuerdo a la posición dada o lo contrario. Esta fórmula se implementará en el algoritmo

de carga cuando se tenga que tomar la decisión final sobre si posicionar un contenedor junto con los demás criterios de escotillas, pesos y posicionamiento.

```

1 public boolean criterioMmpp(int iC, int jC, int kC, int mC, int nC,
  Contenedor con, Barco bar) {
2     Boolean res = false;
3     int criba = 0;
4     if (con.getMmpp().getUn_1() == 0) {//Si no lleva mercancía peligrosa
5         res = true;
6     } else {/*Si lleva mercancía peligrosa. Aplicamos el Diagrama-
  cuestionario del capítulo 7.2 del IMDG. NOTA: CONSIDERANDO BUQUES CON
  ESCOTILLAS ESTANCAS, CON VENTILACIÓN EN BODEGA Y CONTENEDORES CERRADOS.
7         Mmpp candidato = new Mmpp(con.getMmpp().getUn_1(),
  con.getMmpp().getNoex_2(), con.getMmpp().getClase_3(),
  con.getMmpp().getRiesgoSecundario_4(), con.getMmpp().getEmbalaje_5(),
  con.getMmpp().getEys_16a(), con.getMmpp().getEys_16b());
8         localizador:
9         for (int i = 0; i < bar.getMatriz().length; i++) {
10             for (int j = 0; j < bar.getMatriz()[i].length; j++) {
11                 if (bar.getMatriz()[i][j] != null) {
12                     for (int k = 0; k < bar.getMatriz()[i][j].length;
13 k++) {
14                         if (bar.getMatriz()[i][j][k] != null) {
15                             for (int m = 0; m <
16 bar.getMatriz()[i][j][k].length; m++) {
17                                 for (int n = 0; n <
18 bar.getMatriz()[i][j][k][m].length; n++) {
19                                     if
20 (bar.getMatriz()[i][j][k][m][n].getMmpp().getUn_1() != 0) {//Hemos
  encontrado un contenedor con mercancía IMO en el barco
21                                         if
22 (candidato.similarity(candidato.getNoex_2(),
23 bar.getMatriz()[i][j][k][m][n].getMmpp().getNoex_2()) < 0.95) {//Si no son
  misma sustancia (y por lo tanto hace falta segregar)
24                                             String distancia = " ";
25                                             distancia =
26 candidato.segregacion(candidato, bar.getMatriz()[i][j][k][m][n].getMmpp());
27                                             if (distancia == null) {
28                                                 criba++;
29                                                 break localizador;
30                                             } else if
31 (distancia.equals("NO")) {//Analizamos las posiciones relativas entre el
  contenedor que queremos introducir y el que hemos hallado
32                                                 criba++;
33                                                 break localizador;
34                                             } else if
35 (distancia.equals("1")) {Al considerar contenedores cerrados no hay
  restricciones.
36                                                 } else if
37 (distancia.equals("2")) {//Separado de prohíbe que estén en la misma fila a
  no ser que haya separación por cubierta (considero la escotilla como
  separación) Un contenedor de margen en el sentido longitudinal, uno
  transversalmente.
38                                                 int inv = 0;
39                                                 if (k == 0) {

```

```

30                                     inv = 1;
31                                     }
32                                     if ((i == iC && j == jC
&& k == kC && m == mC) || (i == iC && j == jC && k == kC && n == nC) || (i
== iC && j == jC && kC == (k - 1) && n == nC) || (i == iC && j == jC && kC
== (k + 1) && n == nC) || (i == iC && j == jC && l == kC && m == mC && n ==
nC) || (i == iC && j == jC && l == kC && mC == (m + 1) && n == nC) || (i ==
iC && j == jC && l == kC && mC == (m - 1) && n == nC)) {
33                                     criba++;
34                                     break localizador;
35                                     }
36                                     } else if
(distancia.equals("3")) {//Separado de prohíbe que estén en la misma fila a
no ser que haya separación por cubierta (considero la escotilla como
separación) Un contenedor de margen en el sentido longitudinal, dos
transversalmente.
37                                     int inv = 0;
38                                     if (k == 0) {
39                                         inv = 1;
40                                     }
41                                     if ((i == iC && j == jC
&& k == kC && m == mC) || (i == iC && j == jC && k == kC && n == nC) || (i
== iC && j == jC && kC == (k - 1) && n == nC) || (i == iC && j == jC && kC
== (k + 1) && n == nC) || (i == iC && j == jC && l == kC && m == mC && n ==
nC) || (i == iC && j == jC && l == kC && mC == (m + 2) && n == nC) || (i ==
iC && j == jC && l == kC && mC == (m - 2) && n == nC)) {
42                                     criba++;
43                                     break localizador;
44                                     }
45                                     } else if
(distancia.equals("4")) {//Que no compartan bahía.
46                                     if (i == iC) {
47                                         criba++;
48                                         break localizador;
49                                     }
50                                     } else {//Opciones X, casos
especiales, por el momento sin cobertura, solo lo básico. De tener que
desarrollarse se tendría que elaborar el contenido de esta llave.
51                                     criba++;
52                                     break localizador;
53                                     }
54                                     }
55                                     }
56                                     }
57                                     }
58                                     }
59                                     }
60                                     }
61                                     }
62                                     }
63                                     }
64                                     if (criba == 0) {
65                                         res = true;
66                                     }
67                                     return res;
68 }

```

6. INTERFACES Y FUNCIONAMIENTO

En este apartado se presentan los menús del programa, su distribución y tareas.

Cuando se inicia el programa, se abre el menú Nuevo Viaje, en el cual se solicitan los inputs (Barco, ruta y versión de algoritmo de carga):

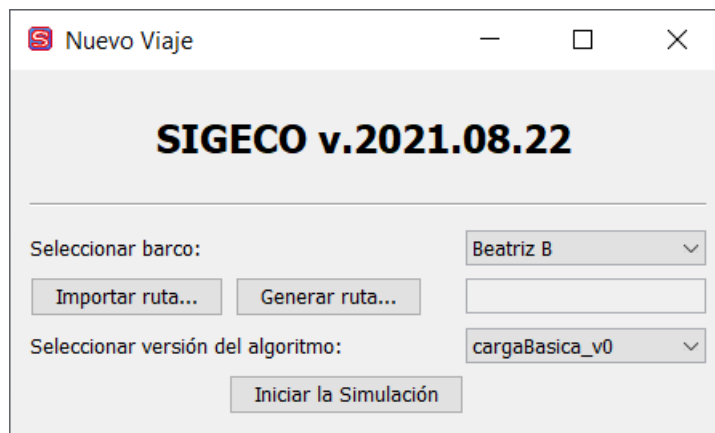


Ilustración 27: Panel para la generación de un viaje. FUENTE: Propia.

Los barcos y los algoritmos ya están prediseñados, las rutas se almacenan en ficheros de texto .rut, los cuales se obtienen por medio del botón importar ruta, que abre el explorador de archivos.

De no haber ruta, se puede pulsar el botón Generar Ruta, que abrirá un asistente para su creación. Una vez creada la ruta se tendrá que añadir por medio de importar ruta.

Las rutas se alojarán por defecto en la siguiente carpeta, C:\Users\usuario\Documents\SIGECO. De no existir, el programa la creará.

El menú de generado de rutas puede accederse por medio en el asistente de nuevo viaje y en el apartado Generar->Ruta del menú principal.

Este asistente se encarga de recopilar los parámetros que introducirá en la fórmula `generadorListaContenedores`, vista en el capítulo 4.4.1. Esta fórmula se ejecutará cuantas cargas tengan que realizarse:

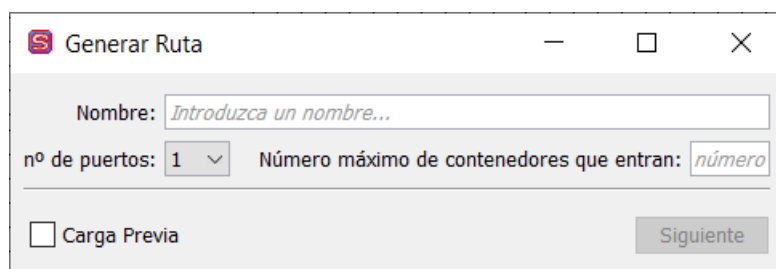


Ilustración 28: Menú generar ruta, parte 1. FUENTE: Propia.

Pongamos una ruta de ejemplo. El barco visitará cuatro puertos, contendrá carga previa y por cada puerto que recorra cargará 5 contenedores con una cobertura por defecto de 0.3, que añadirá un contenedor extra.

La cobertura y la empresa prioritaria son datos que solo se pueden introducir modificando el código. En este ejemplo no hay empresa prioritaria.

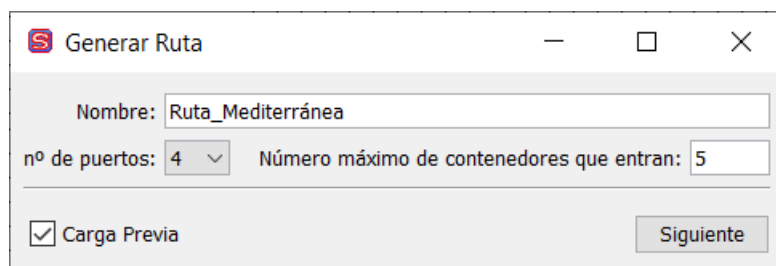


Ilustración 29: Menú generar ruta, parte 1. FUENTE: Propia.

El siguiente cuadro es el de selección de puertos y tiempos. Hay un límite de 11 puertos por ruta.

	Posición original	dd	hh	mm	ss
1º puerto	Abu Dabi	0	0	0	0
2º puerto	Abu Dabi	0	0	0	0
3º puerto	Abu Dabi	0	0	0	0
4º puerto	Abu Dabi	0	0	0	0
5º puerto	Abu Dabi	0	0	0	0
6º puerto	Abu Dabi	0	0	0	0
7º puerto	Abu Dabi	0	0	0	0
8º puerto	Abu Dabi	0	0	0	0
9º puerto	Abu Dabi	0	0	0	0
10º puerto	Abu Dabi	0	0	0	0
11º puerto	Abu Dabi	0	0	0	0

Generar ruta

Ilustración 30: Menú generar ruta, parte 2. FUENTE: Propia.

Cumplimentado quedaría así.

	Posición original	dd	hh	mm	ss
1º puerto	Puerto Saíd	17	6	4	6
2º puerto	Marsaxlokk	1	13	5	55
3º puerto	Gioa Tauro	0	12	30	14
4º puerto	Valencia	1	18	47	35
5º puerto	Abu Dabi	0	0	0	0
6º puerto	Abu Dabi	0	0	0	0
7º puerto	Abu Dabi	0	0	0	0
8º puerto	Abu Dabi	0	0	0	0
9º puerto	Abu Dabi	0	0	0	0
10º puerto	Abu Dabi	0	0	0	0
11º puerto	Abu Dabi	0	0	0	0

Generar ruta

Ilustración 31: Menú generar ruta, parte 2. FUENTE: Propia.

Al pulsar el botón “Generar ruta”, se creará un fichero en una carpeta nominada, en este caso, la ruta generada (pasada por `toStringInforme`) fue la siguiente:

Ruta_Mediterránea

Con carga previa

```
Contenedor _0_n1 = new Contenedor(0, 1, "EMCU3235109", "45G0", lis.SOMGQ, lis.MTMAR, 3900.0, 28600.0, 5496.03, 9396.0, lisM.MMPP_0, lis.EMC);
Contenedor _0_n2 = new Contenedor(0, 2, "HBLU5573587", "42U2", lis.IDJKT, lis.ITGIT, 3680.0, 28320.0, 24832.7, 28513.0, lisM.MMPP_0, lis.HBL);
Contenedor _0_n3 = new Contenedor(0, 3, "SOIU7109582", "L5G1", lis.SGSIN, lis.MTMAR, 4260.0, 29720.0, 25221.98, 29482.0, lisM.MMPP_0, lis.SOI);
Contenedor _0_n4 = new Contenedor(0, 4, "TTNU0621285", "45G3", lis.ESLPA, lis.ITGIT, 3900.0, 28600.0, 25872.0, 29772.0, lisM.MMPP_0, lis.TTN);
Contenedor _0_n5 = new Contenedor(0, 5, "MSCU3218224", "42G0", lis.FRLEH, lis.ESVLC, 3750.0, 28720.0, 5582.39, 9332.0, lisM.MMPP_0, lis.MSC);
Contenedor _0_n6 = new Contenedor(0, 6, "TEXU7508564", "45G0", lis.GBSOU, lis.ESVLC, 3900.0, 28600.0, 1633.41, 5533.0, lisM.MMPP_3397, lis.TEX);
```

Con la siguiente ruta:

1º carga en el puerto de Puerto Saíd (Egipto) --EGPSD--

Al que se llegó tras un viaje de 6h 4m 6s (17)

```
Contenedor _1_n1 = new Contenedor(1, 1, "TTNU0103497", "42P1", lis.EGPSD, lis.ESVLC, 5000.0, 35000.0, 13946.05, 18946.0, lisM.MMPP_0, lis.TTN);
Contenedor _1_n2 = new Contenedor(1, 2, "MSCU6956582", "42G0", lis.EGPSD, lis.ITGIT, 3750.0, 28720.0, 13114.65, 16865.0, lisM.MMPP_0, lis.MSC);
Contenedor _1_n3 = new Contenedor(1, 3, "CSNU1176102", "25G1", lis.EGPSD, lis.MTMAR, 2340.0, 28180.0, 1326.66, 3667.0, lisM.MMPP_1759, lis.CSN);
Contenedor _1_n4 = new Contenedor(1, 4, "TEXU8183801", "22V0", lis.EGPSD, lis.ITGIT, 2300.0, 25000.0, 10868.39, 13168.0, lisM.MMPP_0, lis.TEX);
Contenedor _1_n5 = new Contenedor(1, 5, "ZGRU7344444", "42G0", lis.EGPSD, lis.MTMAR, 3750.0, 28720.0, 3319.41, 7069.0, lisM.MMPP_0, lis.ZGR);
Contenedor _1_n6 = new Contenedor(1, 6, "CMAU0966787", "25R0", lis.EGPSD, lis.ESVLC, 3080.0, 24320.0, 3885.76, 6966.0, lisM.MMPP_0, lis.CMA);
```

2º carga en el puerto de Marsaxlokk (Malta) --MTMAR--

Al que se llegó tras un viaje de 13h 5m 55s (1)

```
Contenedor _2_n1 = new Contenedor(2, 1, "YMLU3233709", "22B3", lis.MTMAR, lis.ESVLC, 2300.0, 25000.0, 13815.41, 16115.0, lisM.MMPP_0, lis.YML);
Contenedor _2_n2 = new Contenedor(2, 2, "THIU6853003", "45G1", lis.MTMAR, lis.ITGIT, 3900.0, 28600.0, 15898.72, 19799.0, lisM.MMPP_0, lis.THI);
Contenedor _2_n3 = new Contenedor(2, 3, "TTNU3154899", "25R1", lis.MTMAR, lis.ESVLC, 3080.0, 24320.0, 16915.51, 19996.0, lisM.MMPP_0, lis.TTN);
Contenedor _2_n4 = new Contenedor(2, 4, "TLCU1951410", "42G2", lis.MTMAR, lis.ITGIT, 3750.0, 28720.0, 13694.44, 17444.0, lisM.MMPP_0, lis.TLC);
Contenedor _2_n5 = new Contenedor(2, 5, "MSCU1635126", "L5V0", lis.MTMAR, lis.ITGIT, 4260.0, 29720.0, 13107.62, 17368.0, lisM.MMPP_0, lis.MSC);
Contenedor _2_n6 = new Contenedor(2, 6, "YMLU9675673", "42G1", lis.MTMAR, lis.ITGIT, 3750.0, 28720.0, 22111.87, 25862.0, lisM.MMPP_0, lis.YML);
```

3º carga en el puerto de Gioia Tauro (Italia) --ITGIT--

Al que se llegó tras un viaje de 12h 30m 14s (0)

```
Contenedor _3_n1 = new Contenedor(3, 1, "TEXU2214994", "45V0", lis.ITGIT, lis.ESVLC, 3900.0, 28600.0, 10514.2, 14414.0, lisM.MMPP_0, lis.TEX);
Contenedor _3_n2 = new Contenedor(3, 2, "MSCU4872902", "25G0", lis.ITGIT, lis.ESVLC, 2340.0, 28180.0, 15195.96, 17536.0, lisM.MMPP_0, lis.MSC);
Contenedor _3_n3 = new Contenedor(3, 3, "CMAU0759239", "25V3", lis.ITGIT, lis.ESVLC, 2340.0, 28180.0, 27227.44, 29567.0, lisM.MMPP_2077, lis.CMA);
Contenedor _3_n4 = new Contenedor(3, 4, "TTNU1979308", "42V3", lis.ITGIT, lis.ESVLC, 3750.0, 28720.0, 22869.84, 26620.0, lisM.MMPP_0, lis.TTN);
Contenedor _3_n5 = new Contenedor(3, 5, "RWTU0428102", "42G2", lis.ITGIT, lis.ESVLC, 3750.0, 28720.0, 15945.46, 19695.0, lisM.MMPP_0, lis.RWT);
Contenedor _3_n6 = new Contenedor(3, 6, "MSCU1614329", "42G2", lis.ITGIT, lis.ESVLC, 3750.0, 28720.0, 25779.42, 29529.0, lisM.MMPP_0, lis.MSC);
```

La interfaz principal tiene la siguiente disposición:

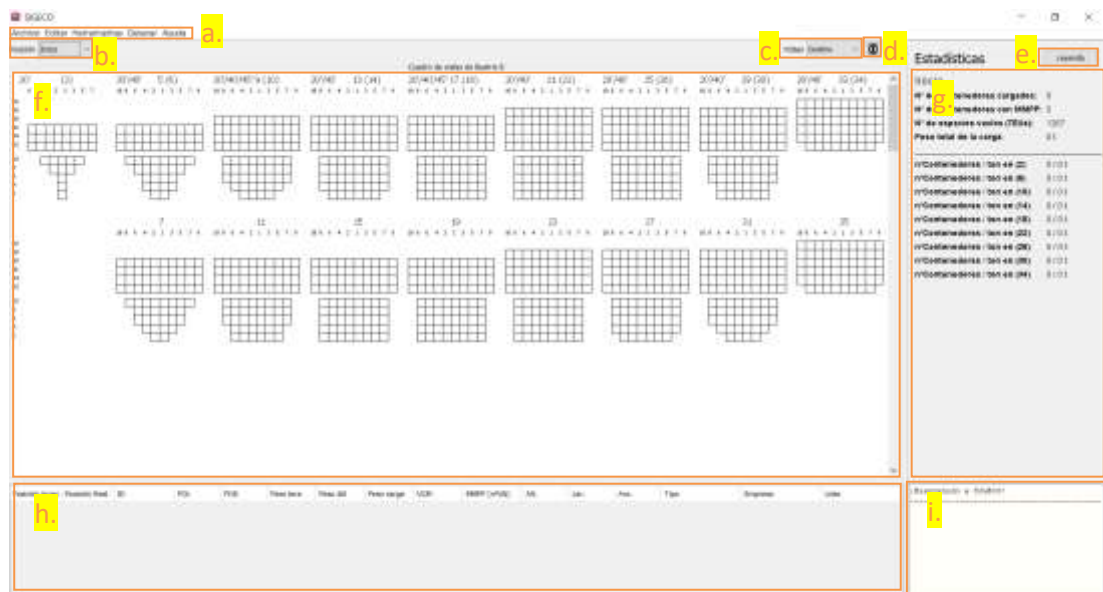


Ilustración 32: Interfaz principal de SIGECO. FUENTE: Propia.

Donde cada área marcada representa lo siguiente:

- a. Barra de menú. Decorativo con vistas a dar una idea del formato final del programa. Solo tiene funcionalidad Generar->Ruta.
- b. Selector de posición del barco. Es un desplegable en el cual se puede seleccionar la situación de la carga dado un punto de la ruta.
- c. Selector del coloreado de vistas. Muestra el modo en el que se van a colorear los contenedores.
- d. Selector de modo de visualización. Se alterna entre modo transversal (☐) y modo modo longitudinal (☐).
- e. Botón que alterna el cuadro de estadísticas y el de cuadro de leyenda.
- f. Visor del plano de carga.
- g. Panel de cuadro de estadísticas o de leyenda.
- h. Tabla de contenedores cargados.
- i. Log.

7. APLICACIÓN PRÁCTICA DE SIGECO

Como se planteó en los objetivos al principio de este trabajo, queríamos crear un programa “global” que pudiese trabajar con distintos barcos, rutas y algoritmos.

En estos últimos tres ejemplos vamos a trabajar con el Beatriz B y otros dos barcos con diferentes tamaños y bodegas. Se generarán rutas para cada uno de los casos, con diferentes puertos y tamaños de packing list.

Ejemplo: Muestra del funcionamiento los algoritmos

En este apartado vamos a introducir una ruta muy sencilla (Rotterdam, Nueva York, Miami) con packing list de 800 contenedores para el barco Beatriz B. Con la particularidad de que vamos a utilizar los algoritmos de carga 0 y 3. Las capacidades de cada uno vienen dadas por la tabla del capítulo 5.1

APLICACIÓN DE cargaBasica v0 (MODO POD)

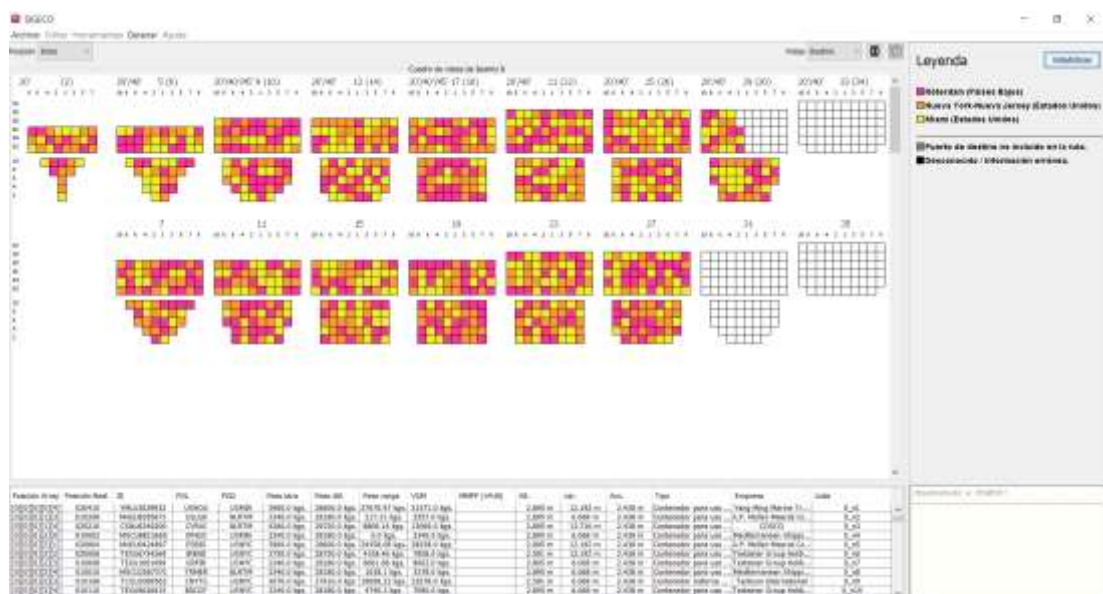


Ilustración 33: SIGECO para Beatriz B con cargaBasica v0. FUENTE: Propia.

APLICACIÓN DE [cargaBasica v0](#) (MODO VGM)



Ilustración 34: SIGECO para Beatriz B con cargaBasica_v0. FUENTE: Propia.

APLICACIÓN DE [cargaBasica v0](#) (MODO DIMENSIÓN)



Ilustración 35: SIGECO para Beatriz B con cargaBasica_v0. FUENTE: Propia.

Como puede observarse en las imágenes, este algoritmo se limita a introducir contenedores sin atenerse a criterios.

APLICACIÓN DE [cargaBasica v3](#) (MODO POD)



Ilustración 36: SIGECO para Beatriz B con cargaBasica_v3. FUENTE: Propia.

APLICACIÓN DE cargaBasica v3 (MODO VGM)

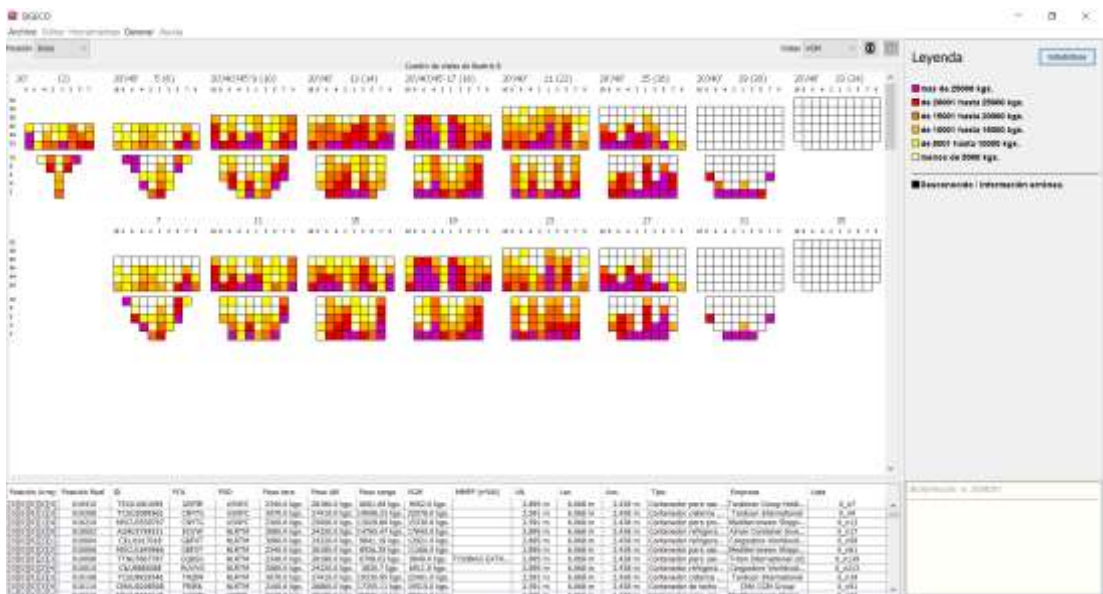


Ilustración 37: SIGECO para Beatriz B con cargaBasica_v3. FUENTE: Propia.

APLICACIÓN DE [cargaBasica_v3](#) (MODULO DIMENSIÓN)



Ilustración 38: SIGECO para Beatriz B con cargaBasica_v3. FUENTE: Propia.

Como se puede comparar, en esta versión ya estaban en base a puertos, pesos y dimensiones.

Ejemplo A: Beatriz B y una ruta Mediterránea.



Ilustración 39: El Buque Beatriz B. FUENTE: MarineTraffic, AUTOR: Celso Hernández.

En el siguiente ejemplo cargaremos el Beatriz B una ruta entre Ambarli, Esmirna, El Pireo, Marsaxlokk, Gioia Tauro, Génova, Barcelona, Valencia y

Algeiras. En cada puerto habrá un pl 1200 contenedores. Hay carga previa.
Se usará el algoritmo de carga `cargaBasica v06`.

Situación Inicial (MODO POD)

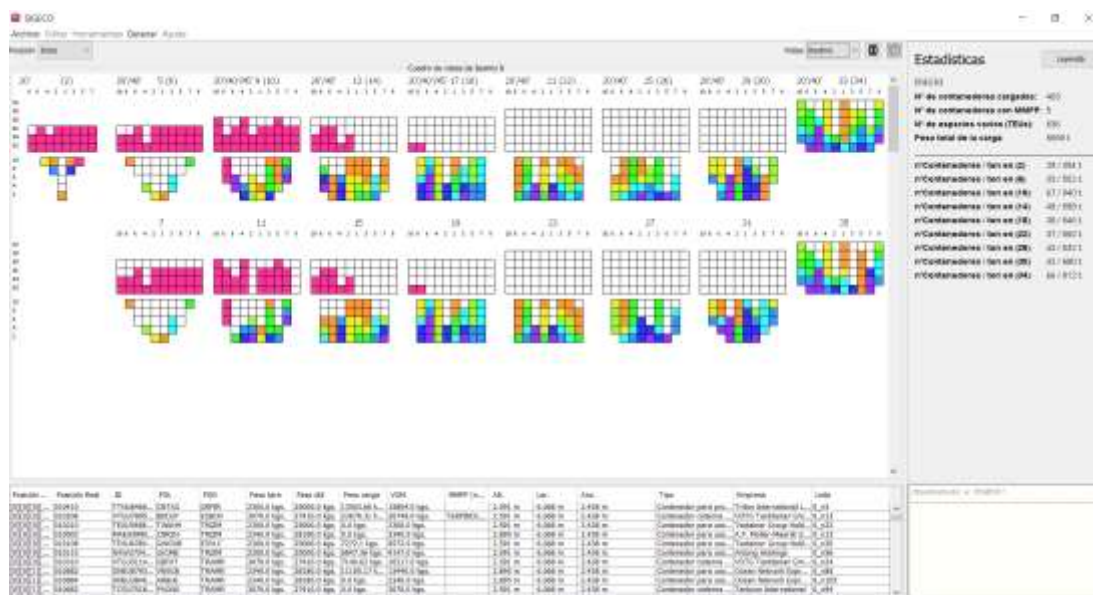


Ilustración 40: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Situación en Marsaxlokk (MODO POD)

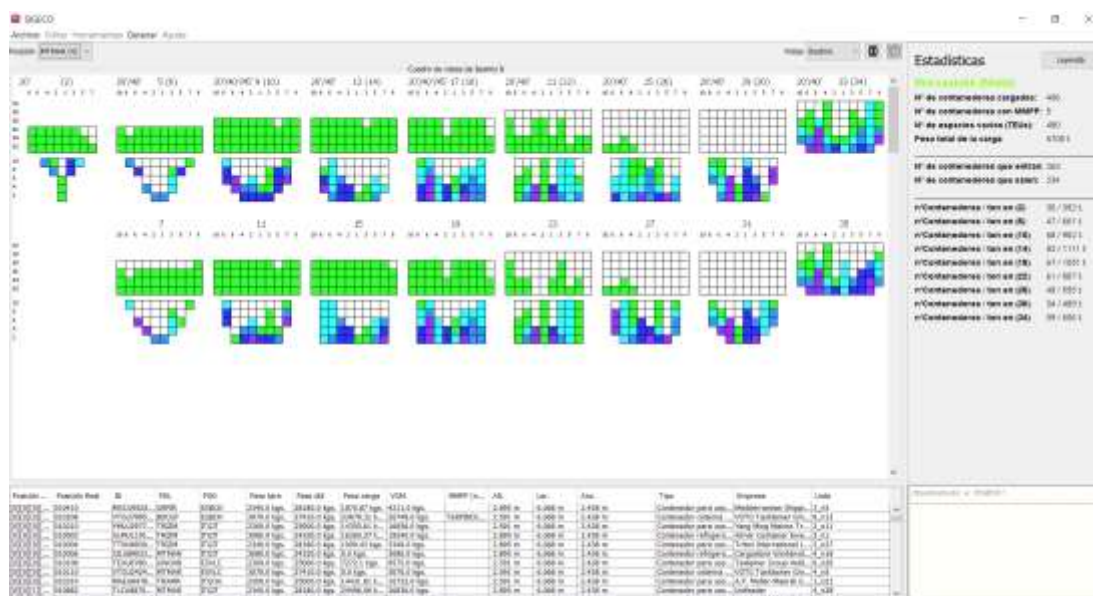


Ilustración 41: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Situación en Marsaxlokk (MODO VGM)



Ilustración 42: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Situación en Marsaxlokk (MODO DIMENSIÓN)



Ilustración 43: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Situación en Barcelona (MODO DIMENSIÓN Y L_c PARA LA FILA 2)

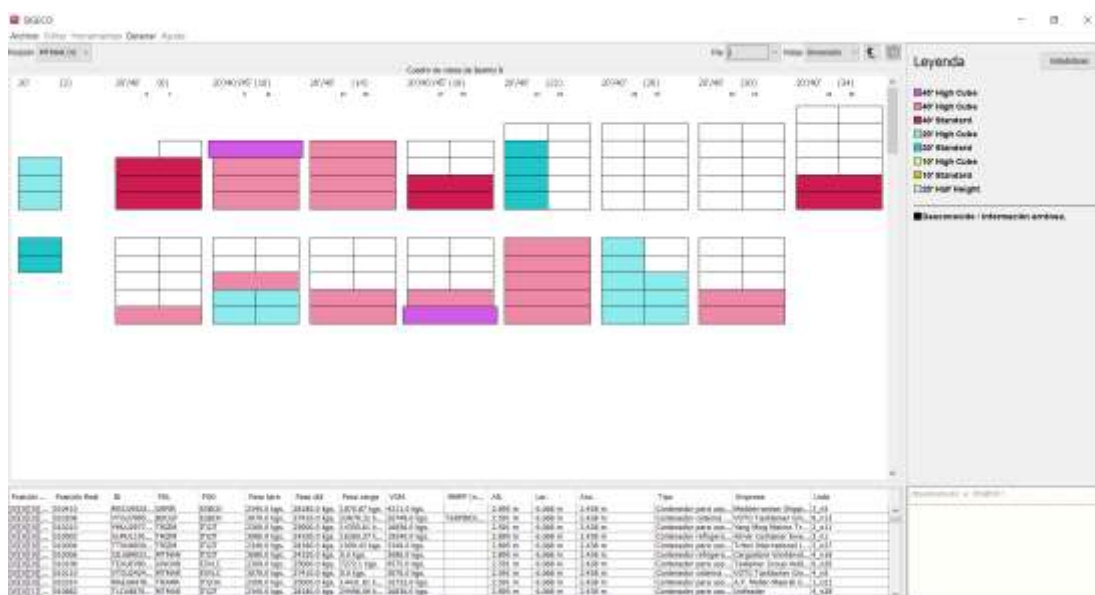


Ilustración 44: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Situación en Valencia (MODO POD)



Ilustración 45: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Situación en Valencia (MODO POD)



Ilustración 46: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Situación en Barcelona (MODO TIPO)



Ilustración 47: SIGECO para Beatriz B y ruta Mediterránea. FUENTE: Propia.

Ejemplo B: MSC Positano y una ruta por África.



Ilustración 48: El buque MSC Positano. FUENTE: MarineTraffic. AUTOR: David Leonard.

En el segundo ejemplo cargaremos el buque MSC Positano por una ruta entre Tenerife, Malabo, Bata, Malabo otra vez, Nouachott y retorno a Tenerife. En cada puerto habrá un pl de 1200 contenedores. Hay carga previa. Se usará el algoritmo de carga `cargaBasica_v06`.

Situación en Tenerife (MODO POD)



Ilustración 49: SIGECO para MSC Positano y ruta Africana. FUENTE: Propia.

Situación en Malabo (MODO POD)



Ilustración 50: SIGECO para MSC Positano y ruta Africana. FUENTE: Propia.

Situación en Bata (MODO DIMENSIÓN)



Ilustración 51: SIGECO para MSC Positano y ruta Africana. FUENTE: Propia.

Situación en Malabo (MODO VGM)



Ilustración 52: SIGECO para MSC Positano y ruta Africana. FUENTE: Propia.

Situación en Malabo (MODO MTY)



Ilustración 53: SIGECO para MSC Positano y ruta Africana. FUENTE: Propia.

Ejemplo C: Caroline Maersk y una ruta entre Europa y Extremo Oriente.



Ilustración 54: El buque Caroline Maersk. FUENTE: MarineTraffic. AUTOR: Sergei Skriabin.

En este último ejemplo el Caroline Maersk visitará Guangzhou, Singapur, Puerto Saíd y Valencia. En cada puerto habrá un pl de 6000 contenedores. Hay carga previa.

Se usará el algoritmo de carga [cargaBasica_v06](#).

Situación Inicial (MODO POD)



Ilustración 55: SIGECO para Caroline Maersk y ruta Asiática. FUENTE: Propia.

Situación en Guangzhou (MODO DIMENSIÓN)



Ilustración 56: SIGECO para Caroline Maersk y ruta Asiática. FUENTE: Propia.

Situación en Puerto Saíd, bahías exclusivas de 40'/45' (MODO DIMENSIÓN)



Ilustración 57: SIGECO para Caroline Maersk y ruta Asiática. FUENTE: Propia.

Situación en Singapur (MODO MMPP)

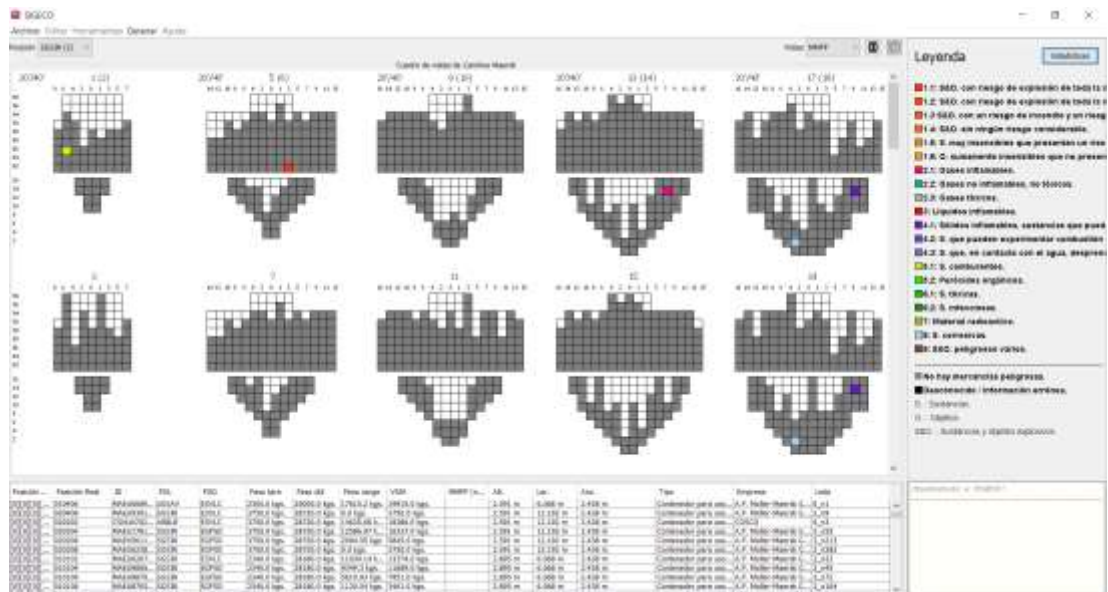


Ilustración 58: SIGECO para Caroline Maersk y ruta Asiática. FUENTE: Propia.

Situación en Puerto Saíd (MODO EMPRESA)



Ilustración 59: SIGECO para Caroline Maersk y ruta Asiática. FUENTE: Propia.

8. OBSERVACIONES DEL PROGRAMA

A lo largo del desarrollo el programa se ha visto puesto a prueba constantemente. Al trabajar con tres variables con infinitos casos es de esperar que haya errores, en especial en las iteraciones más complejas de los algoritmos de carga.

Este programa tiene finalidad académica y no debe utilizarse para aplicaciones reales puesto que los tres barcos introducidos se han hecho en base a planos simplificados de sus bahías y carezco de información sobre sus particularidades, detalles y limitaciones. Un estudio intensivo de su funcionamiento sería necesario.

Con respecto a su fichero ejecutable, habría que ejecutar el programa en varios ordenadores para observar incompatibilidades.

En relación a la generación de contenedores:

- Habría que hacer una investigación sobre si se pudieran cargar listas de contenedores en base a ficheros xml, para que así no todo introducido en ruta fuera generado.
- Una de las principales carencias que tiene la función de generado de contenedores es que conforme se va acercando al último puerto, se van creando contenedores con menor variedad de POD, hasta en la penúltima visita donde todos los contenedores van a ser descargados en el puerto siguiente. Esto evidentemente, no es realista, y habría que idear la manera de “continuar rutas” para evitar este defecto.
- Se podrían simular contenedores Out of Gauge, creando un contenedor especial “OOG” que sería un híbrido entre un contenedor hueco y un contenedor, y simularía el espacio que la extradimensión limitaría en un TEU.
- De forma análoga a los contenedores OOG, se podrían crear conjuntos de contenedores que trabajaran “en bloque” para simular una cama de flats sobre la que se posicionase mercancía break bulk.

En relación a los barcos:

- En las últimas semanas antes de presentar este trabajo llegué a la conclusión de que, pese a simplificar las cosas, no es una buena idea hacer la matriz del barco directamente con objetos contenedor. Habría que crear una clase nueva “Posición” que tuviese asociada:
 - Sus coordenadas para que el programa fuese más eficiente y no tuviese que estar haciendo conversiones todo el tiempo.
 - Un atributo contenedor que por defecto sería hueco o estructural. Este es el único valor que se modificaría.
 - Una serie de atributos de texto que indicarían particularidades de esta posición, como que estuviese reservada para contenedores refrigerados.
- Se considera que las escotillas son únicas para toda la sección de bahíaDupla, lo cual no tiene por que corresponderse con la realidad.

En relación a los algoritmos de carga:

- Hay que tener en cuenta que las incidencias descritas aquí son en referencia a los siete algoritmos cargaBásica, el programa está estructurado de tal manera que se podrían diseñar algoritmos completamente diferentes sin comprometer el funcionamiento del resto del código.
- Los algoritmos tienden a descartar contenedores con facilidad. Esto se debe a que trabajan los listados siguiendo un orden fijo dado por cómo fueron generados. La solución a esta problemática podría ser que antes de proceder a las cargas, habría que reordenar los packing lists dando prioridad a los contenedores más restrictivos, como los que salen en los últimos puertos o los que son muy pesados.
- Los algoritmos hacen una distinción entre la carga previa y el resto de las cargas que duplica las líneas de código. Esto habría que simplificarlo pues dificulta la lectura y las labores de corrección.
- Una de las lecciones mas importantes ha sido en relación con la

estiba en base a puertos en destino. El programa tiene que estudiar la posibilidad de que el barco visite varias veces el mismo puerto y esto supone que no podemos hacer un comparador sencillo entre un pod y un listado de puertos en ruta cuando está trabajando un algoritmo de carga. Ya que hay que estudiar las posiciones relativas del barco y los puertos repetidos restantes. En conclusión, si hay que enumerar los puertos, que sea con caracteres numéricos u objetos no repetibles. Como, por ejemplo, un objeto que combine un puerto con una fecha, y además así la clase fecha ganaría utilidad.

- El funcionamiento de las escotillas se limita al cargar en cubierta los contenedores que salen en el puerto siguiente.
- Los algoritmos no intentan agrupar los contenedores de un mismo. POD en una misma bahía.
- La segregación de mercancías peligrosas no tiene en cuenta las disposiciones y está sujeta a muchas restricciones. Habría que someterla a un mayor estudio de casos.

9. CONCLUSIONES

Tras ver la problemática de los programas de estiba, creo que se ha conseguido hacer una simulación que se ciñe a los requerimientos básicos que se podrían exigir a este tipo de códigos.

Cuenta con dos grandes ventajas: En primer lugar, de seguir trabajando en su desarrollo cuenta con un gran potencial ya que cuenta con una buena base de métodos y clases de soporte. Y en segundo lugar tiene la particularidad de contar con algoritmos que están diseñados para poder trabajar con barcos y rutas independientemente de su tamaño y complejidad.

Quedo particularmente satisfecho con la sencillez con la que funciona, simplemente hay que introducir los parámetros y recibir informes con resultados.

ANEXO I: INDICACIONES SOBRE JAVA

Para aquellas personas que no estén familiarizadas con Java, voy a realizar en este apartado una rápida introducción sobre este lenguaje de programación en el que se proporcionarán las nociones básicas para comprender su funcionamiento.

Un ordenador entiende solamente el lenguaje binario (unos y ceros), sin embargo, las personas somos incapaces de leer e interpretar estas secuencias. Para poder transmitir instrucciones al ordenador se utiliza un lenguaje de programación, que es un código que las personas podemos entender y escribir y que a su vez se puede “traducir” a binario por medio de un proceso llamado compilación, así el ordenador puede comprender nuestras instrucciones.

Dentro de un programa de Java encontramos un primer nivel de organización llamado paquetes y estos a su vez tienen asignados una serie de ficheros de texto llamados clases, que es donde se escribe el código. El programador adjudica a estas clases un cometido concreto y después se las hace trabajar en conjunto: Hay clases que almacenan información, otras declaran objetos, otras se utilizan para la ejecución del programa en sí.

En la clase se escriben variables o métodos. Las primeras son “unidades de información” compuestas por un identificador y el tipo de valor que almacena. Pueden ser:

- Un valor numérico: Entero (`int`), o real (`double`). Ejemplos: `3.4`, `21`.
- Un valor carácter (`char`). Ejemplos: `'g'`, `'@'`, `'4'`.

Un valor lógico (`boolean`). Ejemplos: `true` o `false`.

Los valores pueden ser unitarios o en conjuntos “arrays” de una o varias dimensiones de una misma variable. Para indicarlos se usan corchetes (`[]`) después de declarar el tipo de valor, por ejemplo: Para un conjunto de números enteros de dos dimensiones se mostraría como `int[][]`, una cadena de texto será `char[]`, implementado como objeto, se llama `String`.

La declaración de una variable se utiliza para asignar a esta un espacio de memoria. Para ello se indica su valor, su tipo y se le asocia un nombre. Por ejemplo, la gravedad se declararía como:

```
1 double grav = 9.80665;
```

Donde `double` es el tipo de valor, `grav` su identificador, `(=)` es el símbolo de asignación, `9.80665` es el valor y el punto y coma `(;)` es un símbolo que indica al compilador el cambio de línea.

Las variables pueden manipularse con los operadores, hay tres tipos:

- Operadores matemáticos:
 - `(+)` para la suma.
 - `(-)` para la resta.
 - `(*)` para el producto.
 - `(/)` para la división.
 - `(%)` para el resto.
- Operadores lógicos:
 - `(&&)` operador AND.
 - `(||)` operador OR.
 - `(!)` operador NOT.
- Operadores de relación:
 - `(==)` para la igualdad.
 - `(!=)` para la desigualdad.
 - `(>)` mayor que.
 - `(<)` menor que.
 - `(>=)` mayor igual que.
 - `(<=)` menor igual que.

Se pueden insertar comentarios en el código con los símbolos `(//)` para comentar una línea, o `(/* */)` para comentar párrafos. Estos símbolos actúan como aviso al compilador para que ignore el texto que contienen.

Los métodos son los algoritmos, procedimientos o estructuras responsables de ejecutar los cálculos. Vienen definidos por los valores de entrada y de

salida de éstos. Después de su declaración contienen entre llaves ({}) sus cálculos. Por ejemplo, un método para el cálculo del peso de un objeto:

```
1 double calculoW(double masa) {  
2     return masa * grav;  
3 }
```

En este método llamado `calculoW` se devolverá (`return`) un número real (primer `double`) resultante del producto entre la constante del ejemplo anterior como `grav` y un parámetro (segundo `double`) llamado `masa`.

Algunas anotaciones:

- Los métodos pueden tener referenciados otros métodos en su interior.

Los métodos que no devuelven ningún dato serán precedidos de `void` y carecen de `return`.

- Se pueden declarar variables dentro de los métodos, pero su alcance dentro del programa quedará reducido al interior de éstos.
- Se pueden utilizar estructuras especiales dentro de los métodos, las más importantes son:
 - Estructuras `if(esto){caso1} else{caso2}` en la cual dentro del paréntesis de `if` se analizará una condición lógica (`esto`) y de cumplirse se ejecutará lo que contienen entre sus llaves (`caso1`), de no ser así se ejecutará lo que contiene el segundo caso (`caso2`), la introducción de `else` es opcional.
 - Estructuras `if(esto1){caso1} else if(esto2){caso2} else{caso3}` similar a la anterior pero con una segunda opción de no cumplirse la condición del primer `if`, se pueden concatenar más cadenas `else if` de ser necesario.
 - Estructuras en bucle, las cuales se ejecutarán una vez tras otra mientras se cumpla alguna condición lógica, los más habituales son `for(contador,condición,cambioCondición){caso}` que representa “para” y `while(condición){caso y cambioCondición}`, que representa “mientras”.
- El método encargado de la ejecución del código es el `main`, es único del programa.

Java es un lenguaje orientado a objetos en el cual en una clase se declara un “molde” de variables, los atributos, para poder trabajar con todo ese conjunto. La declaración de un objeto tiene las siguientes partes:

- Declaración de atributos.
- Métodos básicos.
 - Constructor, son métodos especiales que dan a entender al programa que queremos tratar los datos como si de una unidad se tratara.
 - Getter y setter, son los métodos que nos dan atributos y modifican al objeto, respectivamente.
 - toString, para obtener la información del objeto en forma de texto.
- Métodos secundarios, que son todas las funciones adicionales.

Los getter, setter y toString, al ser tan evidentes y similares entre ellos, se tenderán a ignorar a lo largo del trabajo.

Ilustremos esto con un ejemplo:

Queremos tener un registro con datos personales de la tripulación de un barco. En él queremos tener el nombre (`String`), primer apellido (`String`), género (`char`), edad (`int`), altura (`double`) y puesto (`String`). Primero crearemos una clase que agrupe estos datos, que llamaremos `Tripulante` y en ella declaramos los datos del molde que va a ser para almacenar los datos.

```
1 String nombre;  
2 String apellido;  
3 char sexo,  
4 int edad;  
5 double altura;  
6 String puesto;
```

Creamos el constructor:

```
1 Tripulante () {}  
2  
3 Tripulante (String n, String a, Char s, Int e, double a, String p){  
4     this.nombre = n;  
5     this.apellido = a;  
6     this.sexo = s;  
7     this.edad = e;  
8     this.altura = a;  
9     this.puesto = p;  
10 }
```

Creamos los getter y los setter. Ejemplo para la variable nombre:

```
1 String getNombre() {  
2     return nombre;  
3 }
```

```
1 void setNombre(String n) {  
2     this.nombre = n;  
3 }
```

Creamos un método toString:

```
1 String toString() {  
2 String texto = "Tripulante: "+this.nombre+ " " +this.apellido+ ", ";  
3 if(this.sexo=='m') {  
4     texto = texto + "varón, ";  
5 } else {  
6     texto = texto + "mujer, ";  
7 }  
8 return texto + this.edad + " años, " + this.altura + "m, " + this.puesto +  
9     ". /n";  
9 }
```

Los datos de los tripulantes podrán declararse así:

```
1 Tripulante tripl = new Tripulante ("Félix","Pérez",'m',37,1.73,"1°  
Máquinas");
```

Que si se pasa por tripl.toString enviará por consola el siguiente texto:

```
Tripulante: Félix Pérez, varón, 37 años, 1.73 m, 1° Máquinas.
```

De necesitarse métodos extra, una vez declarados los básicos, podrían añadirse a la clase creada todos los algoritmos que hagan falta.

Una vez creado un objeto éste puede ser utilizado para construir otros objetos a la vez, de esta manera podemos crear estructuras virtuales que contengan una gran variedad de datos con los que poder trabajar y que el programa sea capaz de reconocer.

Por ejemplo, la siguiente clase, Equipo, contiene un nombre del barco (String), número de tripulantes (int), edad media de la tripulación (double) y un conjunto de tripulantes (Tripulante[]), en base al objeto creado en el ejemplo anterior.

En un fichero aparte, la disposición de las variables quedaría así:

```
1 String nombreBarco;
2 int numTripulantes;
3 double edadMedia;
4 Tripulante[] gente;
```

El constructor en este caso solo introducirá los datos de `nombreBarco` y `gente`, puesto que los otros dos datos vienen condicionados por el array de tripulantes.

El constructor sería el siguiente:

```
1 Equipo () {}
2
3 Equipo (String nBar, Tripulante[] gente){
4     this.nombreBarco = nBar;
5     this.numTripulantes = gente.length;
6     int media = 0;
7     for(int i=0;i<gente.length;i++){
8         media = media + gente[i];
9     }
10    this.edadMedia = media/gente.length;
11    this.equipo = gente;
12 }
```

Creamos los getter y los setter. Ejemplo para la variable `equipo`:

```
1 Tripulante[] getGente() {
2     return gente;
3 }
```

```
1 void setGente(Tripulante[] gente) {
2     this.gente = gente;
3 }
```

Creamos un método `toString`:

```
1 String toString(){
2 String texto = "La tripulación del barco " +this.nombreBarco+ ", lo componen
" +this.numTripulantes+ ", con una media de edad de " +this.edadMedia+ "
años, está compuesta por: /n";
3     for(int i; i<gente.length;i++){
4         texto = gente[i].toString();
5     }
6     return texto;
7 }
```

Se crea un listado de objetos que compondrán el array de tripulantes `tripSomo`. Este array estará contenido dentro de la clase `Equipo`.

Dados los siguientes datos para un barco llamado Somo:

```
1 Tripulante trip2 = new Tripulante ("Begoña", "Ibáñez", 'f', 35, 1.73, "1°  
Oficial");  
2 Tripulante trip3= new Tripulante ("Valeria", "Sanabria", 'f', 27, 1.67,  
"Camarera");  
3 Tripulante trip4 = new Tripulante ("Antón", "López", 'm', 23, 1.70,  
"Marinero");  
4 Tripulante trip5 = new Tripulante ("Víctor", "Santamaría", 'm', 34, 1.68,  
"Marinero");  
5 Tripulante trip6 = new Tripulante ("Eulalio", "Fresneda", 'm', 53, 1.76,  
"Capitán");  
6 Tripulante[] tripSomo = {trip1, trip2, trip3, trip4, trip5, trip6};  
7 Equipo e1 = new Equipo("Somo", tripSomo);
```

En el main, se introducirá e1.toString:

```
La tripulación del barco Somo, lo componen 6 personas, con una media de edad de  
34,6666667 años, está compuesta por:  
Tripulante: Félix Pérez, varón, 37 años, 1.73 m, 1° Máquinas.  
Tripulante Begoña Ibáñez, mujer, 35, 1.73, 1° Oficial.  
Tripulante: Valeria Sanabria, mujer, 27 años, 1.67 m, Camarera.  
Tripulante: Antón López, varón, 23 años, 1.70 m, Marinero.  
Tripulante: Víctor Santamaría, varón, 34, 1.68, Marinero.  
Tripulante: Eulalio Fresneda, varón, 53, 1.76, Capitán.
```

Desde la propia Oracle, la empresa encargada de desarrollar Java, otorga al programador una serie de librerías con código ya escrito. De esta manera, no es necesario por ejemplo escribir desde cero los métodos necesarios para la elaboración de funciones trigonométricas o la creación de interfaces, agilizando mucho la velocidad del trabajo y brindando al programador la garantía de estar trabajando con código elaborado por expertos. Toda librería necesitada para la elaboración de las clases básicas es mencionada.

ANEXO II: AVISO RESPONSABILIDAD UC

AVISO:

Este documento es el resultado del Trabajo Fin de Grado de un alumno, siendo su autor responsable de su contenido.

Se trata por tanto de un trabajo académico que puede contener errores detectados por el tribunal y que pueden no haber sido corregidos por el autor en la presente edición.

Debido a dicha orientación académica no debe hacerse un uso profesional de su contenido.

Este tipo de trabajos, junto con su defensa, pueden haber obtenido una nota que oscila entre 5 y 10 puntos, por lo que la calidad y el número de errores que puedan contener difieren en gran medida entre unos trabajos y otros,

La Universidad de Cantabria, la Escuela Técnica Superior de Náutica, los miembros del Tribunal de Trabajos Fin de Grado así como el profesor tutor/director no son responsables del contenido último de este Trabajo.”

BIBLIOGRAFÍA

UNE Normalización Española (2013) 'UNE-EN ISO 6346:1996/A3:2013', *UNE Normalización Española*. Available at: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0051406>.

Maersk (2021) *Emma Maersk / Container vessel of Maersk shipping line*. Available at: <http://www.emma-maersk.com/>.

The Maritime Executive (2021) 'Largest Containership Arrives in Taiwan on Maiden Voyage', *The Maritime Executive*, 9 August. Available at: <https://maritime-executive.com/article/largest-containership-arrives-in-taiwan-on-maiden-voyage>.

IMO (2014) *IMDG Code.international maritime dangerous goods code.*, *IMO Publishing*.

Carpenter, M. A. (2006) 'The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger. Marc Levinson', *Administrative Science Quarterly*. doi: 10.2189/asqu.51.4.656.

Burgos, J. M. and Galve, J. (2014) *Programación I*. Pearson.

StackOverflow (2017) *Similarity String Comparison in Java - Stack Overflow*. Available at: <https://stackoverflow.com/questions/955110/similarity-string-comparison-in-java?form=MY01SV&OCID=MY01SV>.

Oracle (2020) *Overview (Java Platform SE 7)*. Available at: <https://docs.oracle.com/javase/7/docs/api/overview-summary.html>.

Container ship cargo stowage and planning procedures (2015). Available at: <http://shipsbusiness.com/contcgstwg.html>.

Apache NetBeans (2020) *NetBeans*. Available at: <https://netbeans.apache.org/>.

Planos de los buques Beatriz B, MSC Positano y Caroline Maersk.